Hochschule Heilbronn
Universität Heidelberg
Studiengang Medizinische Informatik

DIPLOMARBEIT

Auswahl und Realisierung antwortadaptiver / respose-adaptiver Randomisationsverfahren für klinische Studien in der Anwendung RANDI2

Verfasser: Natalie Waskowzow
Matrikel-Nr:167055
E-Mail: nwaskowz@stud.hs-heilbronn.de

Referent: Prof. Dr. Martin Haag
Korreferent: Prof. Dr. Wendelin Schramm

Heilbronn, Juli 2011

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben.

Besonders möchte ich mich bei meinem Referent Prof. Dr. Martin Haag und bei meinem Betreuer Daniel Schrimpf für die Betreuung und Unterstützung während dieser Diplomarbeit bedanken.

Großer Dank geht auch an meinen Mann Claudio Morales, der während dieser Zeit immer für mich da war und alles Mögliche gemacht hat, damit ich mich nur um dieser Arbeit kümmern konnte.

Zusammenfassung

Bevor eine neue Therapie zur Medikation zugelassen wird, muss sie in einer klinischen Studie ihren Nutzen beweisen. Da heutzutage die besten Ergebnisse bei solchen Fragestellungen aus randomisierten klinischen Studien kommen, ist eine gute Randomisationssoftware für den Ablauf der Studie von einer großen Bedeutung. In Abhängigkeit von dem Studienaufbau wird ein geeigneter Randomisationsalgorithmus für die Zuweisung der Studienteilnehmer zu den Therapien gewählt.

Ziel dieser Diplomarbeit ist eine schon bestehende Open Source Software RANDI2 zu analysieren und um ein weiteres Randomisationsverfahren zu erweitern. Dieses Verfahren soll zu der Familie der response-adaptiven Randomisationsverfahren gehören. Im Weiteren werden unterschiedliche Randomisationsalgorithmen mit jeweiligen Vorteilen, Nachteilen und Funktionsweisen vorgestellt. Nachdem ein passendes Verfahren gefunden wird, werden die Möglichkeiten untersucht, ihn in schon vorhandene Software zu implementieren. Anschließend wird die Implementierung gemacht und das Ergebnis getestet.

Nach der Erweiterung von RANDI2 soll bei einer response-adaptiven Studie schon während des Ablaufs festgestellt werden können, welche Therapie bessere Ergebnisse erzielt. Mit dieser Kenntnis können mehr Patienten während der Studie eine bessere Behandlung bekommen und früher von dem Nutzen profitieren.

Inhalt

Abbildungsverzeichnis

1. Einleitung

1.1.Gegenstand und Motivation

Ziel klinischer Studien ist der Nachweis der Wirksamkeit und der Vorteile neuer Medikamente oder Therapien gegenüber anderen, schon angewendeten Behandlungsmethoden. Es existieren verschiedene Verfahren für die Durchführung solcher Beweise. Zum Beispiel durch:

- historische Vergleiche,
- prospektive Beobachtungsstudien,
- randomisierte kontrollierte Studien, usw.

Heutzutage werden meistens nur die Ergebnisse von randomisierten kontrollierten Studien als Wirksamkeitsnachweis akzeptiert. Der wichtigste Vorteil dieser Studien ist, dass die Patienten zufällig zur Therapie- und Kontrollgruppe zugeordnet werden. Im einfachsten Fall hat jeder Patient eine bekannte Wahrscheinlichkeit eine der zu testenden Studientherapien zu erhalten. Jeder Patient hat eine gleiche bekannte Wahrscheinlichkeit eine von den zu testenden Studientherapien zu bekommen. Im Idealfall kann man aber nicht vorhersehen, welche Therapie der nächste Patient bekommt. Deswegen wird die
Auswahlverzerrung (Selektions-Bias) ausgeschlossen. Also Verzerrung der Ergebnisse durch Auswahl und Zuordnung von Patienten mit besonders schlechter oder guter Prognose zu den Therapiegruppen.
Es gibt viele und ganz unterschiedliche Arten von Randomisation, so wie:

- einfache Randomisation,
- Blockrandomisation,
- stratifizierte Randomisation,
- Minimisation,
- oder response-adaptive-Randomisation [1].

Welches Verfahren verwendet wird, ist abhängig von der Art und dem Aufbau der Studie.

Die klinischen Studien in der Medizin unterscheiden sich in vielen Hinsichten von randomisierten Experimenten in anderen Bereichen. Der wichtigste Unterschied

klinischer Studien ist die Durchführung am Menschen, dies wirft komplexe ethische Fragen auf, was in anderen wissenschaftlichen Experimenten meist nicht der Fall ist. Solange die Wirksamkeit eines Medikamentes oder einer Therapie nicht bewiesen ist, ist es unmöglich sicher zu sagen, welcher der zu vergleichenden Therapien für den Patienten besser ist. Einerseits kann der Patient die innovative und wirksame Therapie, die noch nirgendwo praktiziert wird, bekommen. Andererseits kann diese Therapie auch großen Schaden für den Patienten verursachen.

Das folgende Beispiel zeigt, warum es wichtig ist, noch während einer Studie auf die erfolgreich abgeschlossenen Fälle zu achten. Im Jahr 1994 publizierte The New England Journal of Medicine eine Studie zur Behandlung von HIV infizierten Frauen während der Schwangerschaft mit dem Zidovudin [2]. Die Kontroll- und die Therapiegruppen waren gleich groß (239 Frauen in Zidovudin-Gruppe und 238 in Placebo). Am Ende der Studie waren 40 neugeborene Kinder von den Frauen aus der Placebo Gruppe HIV positiv, bei Zidovudin-Gruppe waren es nur 13. In diesem Fall war die 50-50 Verteilung der Frauen zu den Gruppen unethisch. Nachdem es zu sehen war, dass in der Zidovudin-Gruppe viel mehr gesunde Kinder zur Welt kommen, sollten mehr Frauen die Zidovudin-Behandlung erhalten.

Verfahren, welche erlauben im Verlauf der Studie mehr Patienten zu der wirksameren Therapie oder Behandlung zuzuordnen, sind response-adaptive-Verfahren. Die Verteilungswahrscheinlichkeiten, die am Anfang der Studie bestimmt wurden, verändern sich entsprechend der Ergebnisse bereits abgeschlossener Behandlungen. So haben mehr Patienten die Chance schon während der Studie eine bessere Behandlung zu bekommen.

1.2. Problemstellung und Ziele

Im Auftrag vom Deutschen Krebsforschungszentrum wird eine Open Source Software (RANDI2[1]) für die Randomisation von klinischen Studien entwickelt. In RANDI2 sind schon mehrere klassische Randomisationsverfahren implementiert und werden erfolgreich angewendet. Die response-adaptive Verfahren wurden bisher nicht umgesetzt.

Jede Studie ist im Prinzip ein wissenschaftliches Experiment. Das führt zu einem ethischen Konflikt, weil es nicht realisierbar ist, dass alle Patienten die bestmögliche Behandlung erhalten und gleichzeitig eine Aussage über die Wirksamkeit der Therapien getroffen werden kann. Da bei response-adaptiven Verfahren die Anzahl erfolgreich behandelter Patienten erhöht wird, ist die Implementierung in RANDI2, vorgesehen. Um das zu erreichen, ist eine Erweiterung von RANDI2 zur Eingabe und Verarbeitung von Verlaufsdaten der Patienten erforderlich.

Es existieren mehrere response-adaptive-Verfahren, mit jeweils unterschiedlichen Einsatzgebieten/Randbedingungen. Die Aufgabe besteht darin, ein geeignetes Verfahren auszuwählen und zu implementieren. Dazu gehört auch die Anwendung RANDI2 dahingehend zu erweitern Verlaufsdaten aufzunehmen, dies sollte möglichst flexibel gestaltet werden, um einen weiteren Ausbau des Programms zu gewährleisten. Weiterhin ist es erforderlich eine einfach zu bedienende Oberfläche zur Eingabe anzubieten, diese soll alle nötigen Daten für den ausgewählten Algorithmus aufnehmen können. Mit diesen Erweiterungen wird es in RANDI2 ermöglicht eine weitere Klasse von Randomisationverfahren anzubieten, deren Fokus auf ethischen Fragestellungen liegt.

[1] http://www.randi2.org/

2. Grundlagen

2.1. RANDI2

„RANDI2“ ist eine webbasierte Open Source Anwendung, die zur Randomisation klinischer Studien dient. Dieses Projekt ist entstanden in einem Kurs („Praktikum Softwareentwicklung“) des Studiengangs „Medizinische Informatik“ (HS Heilbronn/Uni Heidelberg). Das Praktikum hat ein Jahr gedauert und hat am Ende ein sehr gutes Ergebnis gebracht. Nach diesem Erfolg wurde entschieden, diese Arbeit mit der Unterstützung des Deutschen Krebsforschungszentrums fortzusetzen. Daraufhin wurde RANDI2 unter eine Open Source Lizenz gestellt. Um die Anwendung möglichst transparent und für jeden nutzbar weiter zu entwickeln und zu erweitern. Die verwendete Lizenz die „GNU General public license“ (GNU GPL) in der Version 3 ermöglicht das freie Nutzen der Software für jede Gelegenheit bzw. gibt die Freiheit die Quellkode zu verändern und zu teilen.

„RANDI2“ ist mit Hilfe der modernen Technologien und Frameworks (so wie: Hibernate, Spring und ICEFaces) realisiert und unterstützt in aktueller Version 0.7 verschiedene Randomisationsalgorithmen, so wie: Vollständige Randomisation, Schiefer Münzwurf, Trunkierte Randomisation, Blockrandomisation (mit fester und variabler Blocklänge), Minimisation und Wei’s Urnenmodell[2]. Genaueres zu den Randomisationsmethoden wird in Kapitel 2.3.2 erläutert. Es besteht zusätzlich die Möglichkeit, die Studie mit ausgewähltem Verfahren zu simulieren. Mit Hilfe der Simulation des Randomisationsprozesses kann entschieden werden, ob sich ein Verfahren für eine bestimmte Studie eignet.

Genauer über das Projekt, Funktionalitäten und verwendete Technologien kann auf der Homepage www.randi2.org , sowie in Kapitel 3 dieser Arbeit, nachgelesen werden.

[2] www.randi2.org

2.2. Softwaretechniken

2.2.1 Git

Git ist ein verteiltes Versionskontrollsystem, entwickelt von Linus Torvalds um jede Art von Projekten schnell und effektiv zu managen. Git unterscheidet sich in vielen Bereichen von anderen Versionskontrollsystemen (z.B. Apache Subversion (SVN)[3]) und bringt viele Vorteile mit sich [3]:

- Unterstützt verteilte Entwicklung: mehrere Benutzer können aus unterschiedlichen Orten an desselben Projekt arbeiten, auch offline.
- Sehr schnell und performant (sogar bei der Arbeit mit großen Projekten)[4]
- Sicher: um die Objekte zu identifizieren und zu benennen wird das kryptographischer Verfahren (Secure Hash Funktion: SHA1) benutzt. Diese Methode gewährleistet die Integrität und das Vertrauen.
- Erhöht Verantwortung: die Information über jede Veränderung an der Datei wird gespeichert und kann jederzeit verfolgt werden.
- Unterstützt die Entwicklung mit verschiedenen Zweigen
- Vollständige Geschichte der Revision auf jedem Rechner

Obwohl Git ursprünglich für den Linux-Kernel entwickelt wurde, läuft die aktuelle Version auf verschiedenen Betriebssystemen (so wie Microsoft Windows, Mac OS X, Solaris, usw.) Die genaue Einleitung zur Installation und zur Nutzung findet man auf der Wiki-Seite der Homepage[5].

2.2.2 Hibernate

Hibernate ist ein Open Source Framework für Java, das Umgang mit relationalen Datenbanken vereinfacht. Hibernate kümmert sich um das Mapping von Plain Old Java Objects (POJOs) in die Datenbanktabellen und umgekehrt. Das kann dem Entwickler sehr viel Zeit einsparen, da er sich gar nicht mehr darum kümmern muss, wie die Daten in den Java-Objekten im Gegensatz zur relationalen Datenbanken aussehen.

[3] http://subversion.apache.org/

Hibernate kann mit vielen unterschiedlichen Datenbanken betrieben werden. Die aktuelle Version 0.7 der „RANDI2“ unterstützt zum Beispiel gleichzeitig MySQL und PostgreSQL.

Die wichtigsten Vorteile von Hibernate sind[4]:

- Ursprüngliches Programmiermodell: Die persistente Klassen entstehen anhand üblicher Objekt-orientierter Sprache mit Vererbung, Polymorphismus, Assoziationen, Kompositionen und Java Collections Framework.
- Transparente Persistenz: Hibernate braucht keine zusätzlichen Schnittstellen oder Klassen um die Objekte persistent zu speichern. Das Einzige, was gebraucht wird, ist ein parameterloser Konstruktor, damit Hibernate auf die Attribute der Klasse zugreifen kann[5].
- Hohe Performanz
- Sicherheit und Skalierbarkeit

Es gibt eine umfangreiche Bibliothek von Hibernate-Annotationen, die die Entwicklung mit Hibernate sehr einfach und flexibel macht[6].

2.2.3. ICEfaces

„ICEfaces ist ein auf JavaServer Faces (JSF) basierendes Ajax[7]-Framework für Java“[6]. Durch ICEfaces ist es möglich geworden, ohne großen Umstände Ajax-Webanwendungen zu erstellen. Die umfangreiche Komponenten-Bibliothek mit schon eingebauter Ajax-Funktionalität macht dieses Framework aus. Das bedeutet, dass Java-Skript von ICEfaces generiert wird und der Entwickler kein Java-Skript mehr schreiben braucht. Gleichzeitig macht das den Quellcode viel übersichtlicher und besser lesbar.

[4] http://www.hibernate.org/about/why-hibernate

[5] http://de.wikipedia.org/wiki/Hibernate_%28Framework%29

[6] http://www.galileocomputing.de/artikel/gp/artikelID-328

[7] Asynchronous JavaScript and XML

ICEfaces unterstützt das Model-View-Controller Entwurfsmuster, das heißt, dass die Beans (Model), die Oberfläche (View) und die Steuerung (Controller) voneinander getrennt sind [7]. Dieses Konzept gewährleistet höhere Lesbarkeit und einfachere Wartung des Quellcodes.

Auf der Webseite *icefaces.org* werden alle aktuelle Informationen und Neuigkeiten von der ICEsoft Technologies Inc. veröffentlicht.

2.2.4. Spring

Spring ist ein Integrations-Framework für die Java-Plattform. Dieses Framework wurde entwickelt mit dem Ziel, die Java/Java-EE Entwicklung zu vereinfachenS[8].

Spring beinhaltet [9]:

- **Lightweight container:** unterstützt zentrale bzw. automatisierte Konfiguration und Schreiben von Objekten. Dieser Container ermöglicht mehr Agilität der Software.
- **Abstraktionslayer für das Transaktionsmanagement**
- **JDBC Abstraktionslayer:** vereinfacht Fehlersuche und reduziert die Mengen des Quellcodes, die ein Entwickler schreiben soll.
- **Integration von Hibernate**
- **Unterstützt Aspekt-Orientierte Programmierung**
- **Unterstützt Model-View-Controller Entwurfsmuster.**

Spring Framework ist unter Apache Lizenz, Version 2.0 veröffentlicht. Auf der Home-page *http://www.springsource.org/* sind alle aktuelle Informationen und alle Dateien zum Herunterladen zu finden.

2.2.5. JUnit

JUnit ist ein Framework zum Testen von Java-Programmen.
Die Tests werden automatisiert und ohne eine menschliche Intervention durchgeführt. Hauptaufgabe der JUnit-Tests ist die Unterstützung des Entwicklers während der Erstellung des Quelltextes. Idealerweise werden die Tests vor der Implementierung geschrieben und nach der Implementierung ausgeführt. Dies kann dem Entwickler entweder bestätigen, dass die Implementierung so funktioniert, wie es sein soll, oder teilweise das aufwendige Fehlersuche sparen, da ein fehlschlagender Test idealerweise die entsprechende Stelle des Quelltextes identifiziert.

Wie der Name dieses Frameworks zeigt werden die Tests in „Units“ eingruppiert. Jeder „Unit“ ist mit einer Annotation *@Test* gekennzeichnet und haben weder Parameter und noch Rückgabewert. In einem Test wird üblicherweise eine Funktionalität überprüft. In den Methoden mit den Annotationen *@Before* und *@After* können alle notwendigen Vorbereitungen für die Tests bzw. „Aufräumen“ nach den Tests durchgeführt werden. Das Framework sorgt dafür, dass diese Methoden zur richtigen Zeit aufgerufen werden.

Die wichtigste Klasse von JUnit ist die Klasse *Assert*. Mit Hilfe von assert-Methoden, die in dieser Klasse definiert sind, wird der Quelltext in einem Test auf Korrektheit geprüft.

Folgende Klasse *Nachfolger* ist ein Beispiel einer Java-Klasse:

```
public class Nachfolger {
   public int gibNachfolger(int zahl){
       return zahl+1;
   }
}
```

Abbildung 2.2.5. 1: Beispiel einer Java-Klasse

Die JUnit-Testklasse ist auf der Abbildung 2.2.5.2 dargestellt. Diese Klasse beinhaltet zwei spezielle Methoden: *setUp()* und *tearDown()* die mit @Before bzw. @After annotiert sind. Die *setUp()*-Methode wird vor und die *tearDown()* nach jedem Test ausgeführt. Dies ist anhand der beiden @Test-Methoden zu sehen. Die

Methode testGibNachfolger1() läuft fehlerfrei durch. Dagegen testGibNachfolger2() schlägt fehl, da dem Attribut *zahl* nach dem ersten Test in der *tearDown()*-Methode der Wert „0“ und vor dem Test selber in der *setUp()*-Methode der Wert „5“ zugewiesen wird. Mit dem Aufruf der Methode *assertEquals* wird der Wert überprüft.

```
public class NachfolgerTest {

    int zahl = 0;
    Nachfolger nachfolger = new Nachfolger();

    @Before
    public void setUp() throws Exception {
        zahl = 5;
    }

    @After
    public void tearDown() throws Exception {
        if(zahl!=0)
            zahl = 0;
    }

    @Test
    public void testGibNachfolger1() {
        zahl = nachfolger.gibNachfolger(zahl);
        assertEquals(6, zahl);
    }

    @Test
    public void testGibNachfolger2() {
        zahl = nachfolger.gibNachfolger(zahl);
        assertEquals(7, zahl);
    }

}
```

Abbildung 2.2.5. 2: Beispiel einer JUnit-Testklasse

Über alle möglichen Releases und neue Updates, sowie neue Tools, kann man auf der Homepage nähere Informationen holen[8].

[8] www.junit.org

2.3. Randomisationsverfahren

2.3.1.Randomisation in den klinischen Studien

Damit die Wirksamkeit eines neuen Medikamentes vollständig nachgewiesen wird, müssen die vier Phasen der klinischen Studien erfolgreich abgeschlossen werden [10,1].

- Phase I: In dieser Phase wird das Medikament zum ersten Mal an Menschen, überwiegend an gesunden, eingesetzt. Es gibt aber Wirkstoffe, die stark in natürliche Körpervorgänge eingreifen, so wie z.B. Krebsmedikamente. Deswegen werden in diesen Fällen die Medikamente schon in der Phase I meist an schwer kranken Patienten eingesetzt.

  Ziel der Studien dieser Phase ist es Informationen über pharmakokinetische und dynamische Eigenschaften, sowie über Sicherheit und Verträglichkeit der Substanz zu gewinnen. Auch die ersten Nebenwirkungen werden beobachtet.

- Phase II: Hier wird die Behandlung zum ersten Mal an Patienten erprobt. Ziel ist es die richtige Dosierung zu finden, um die optimale Wirkung zu erzielen.

- Phase III: Studien in der Phase III dienen dem Wirksamkeitsnachweis einer neuen Behandlung. Sie sind die Basis für die Zulassung des neuen Medikamentes. Die meisten Studien in dieser Phase sind randomisierte Studien, in denen zwei oder mehr Therapien verglichen werden. Mindestens eine von ihnen ist die innovative Therapie, dessen Wirksamkeit nachgewiesen werden soll. Die andere Behandlungsmöglichkeit (mindestens eine) dient zur Kontrolle. Die Kontrollgruppe kann entweder mit einem Placebo (Scheinarzneimittel), oder einer alternativen, schon bewährten Methode behandelt werden. Die Einteilung zur Therapie- oder Kontrollgruppe erfolgt nach Zufallsprinzip. Deswegen ist die Randomisation eine der zentralen Methoden der Phase III-Studien. Sie schafft die Basis für einen validen Therapievergleich.

- Phase IV: In der Phase IV wird die Sicherheit eines schon zugelassenen Medikamentes weiter überwacht.

In dem Planungsstadium einer klinischen Studie stellt sich die Frage, wie viele Patienten bei der Studie teilnehmen sollen, damit der Unterschied zwischen den

Therapien statistisch nachgewiesen werden kann. Dies ist abhängig von dem wahren - aber zum größten Teil unbekannten - Unterschied zwischen den Behandlungsmethoden. Bei einem großen wahren Unterschied wird eine nicht so große Fallzahl benötigt. Je kleiner der Unterschied ist, desto größer muss die Fallzahl sein. Die Wahrscheinlichkeit einen signifikanten Unterschied zwischen den vergleichenden Therapien zu beobachten heißt Power und sollte im besten Fall so hoch wie möglich sein (80%-90%). Vor dem Start jeder Studie wird die Entscheidung über die Größenverhältnisse der Behandlungsgruppen getroffen. Um die Power zu maximieren, müssen diese Größenverhältnisse während der Studie eingehalten werden [1].

2.3.2. Klassische Randomisationsverfahren

Es gibt verschiedene Techniken der Randomisation [1, 10, 11]. In Abhängigkeit des Studiendesigns werden unterschiedliche Randomisationsverfahren angewendet.

Vollständige Randomisation

Vollständige Randomisation kann man mit einem Münz- oder Würfelwurf vergleichen. Für jeden neuen Patienten, der in die Studie aufgenommen wird, wird einmal die Münze oder der Würfel geworfen. In Abhängigkeit von dem Ausgang des Wurfes wird der Patient zu einer der möglichen Therapien zugewiesen. In der Praxis werden die Sequenzen von Zufallszahlen entweder mit Hilfe von Zufallszahlentabellen oder mit Hilfe von statistischer Software erzeugt. Somit kann die Randomisation noch vor Beginn der Studie abgeschlossen werden. Nachteil dieser Methode ist, dass die Zuordnung von Patienten zur verschiedenen Behandlungsgruppen ungleichmäßig sein kann. Um die Gruppengrößen trotzdem ungefähr gleich oder im geplanten Größenverhältnis zu bekommen, muss mit sehr vielen Studienteilnehmern gerechnet werden.

Blockrandomisation

Mit der Blockrandomisation kann man die Einhaltung des Größenverhältnisses der Behandlungsgruppen unabhängig von der Studiengröße erreichen. Sie garantiert die Ausgeglichenheit in den Gruppen zu jedem Zeitpunkt der Studie. Dafür werden Blöcke von einer festen oder variablen Größe gebildet, in denen die Therapie eine bestimmte Verteilung besitzt. Zum Beispiel wird ein Block von der Größe acht mit vier

Mal eine Therapie und vier Mal die Andere belegt, diese werden zufällig verteilt. Anschließend wird der Block abgearbeitet, was die ausgeglichene Verteilung der Therapiearten gewährleistet. Es muss jedoch beachtet werden, dass wenn die Blocklänge zu kurz gewählt wird oder bekannt ist, besteht die Möglichkeit, dass die nächste Therapie vom Arzt geraten wird, dadurch besteht die Gefahr der Auswahlverzerrung (eng. selection bias). Um das zu vermeiden, können im Verlauf der Studie verschiedene Blocklängen benutzt werden. Dadurch wird das Erraten durch den Arzt erschwert.

Stratifizierte Randomisation

Stratifizierte Randomisation kann die Balance zwischen den Behandlungsgruppen bezüglich der Verteilung prognostisch wichtiger Faktoren gewährleisten, indem für jeden Faktor eine eigene Randomisationsliste erstellt wird. Der Nachteil ist, dass die Anzahl der Randomisationslisten sehr schnell wachsen kann. Zum Beispiel, bei 3 Faktoren mit jeweils 2 Ausprägungen braucht man 8 Listen(2^3). Deshalb sollten nur wenige Schichtungskriterien gewählt werden. Die stratifizierte Randomisation macht die Studienergebnisse aussagekräftiger und minimiert die Verzerrungen. Zum Beispiel wird bei der unstratifizierten Blockrandomisation nur die Einhaltung des Größenverhältnisses der Behandlungsgruppen erreicht. Bei der stratifizierten Blockrandomisation wird aber auch das Gleichgewicht in der Verteilung prognostisch wichtiger Faktoren gewährleistet.

Minimisation

Minimisation ist eine Alternative zur stratifizierten Randomisation. Bei diesem Verfahren werden die Ausprägungsstufen der einzelnen prognostischen Faktoren betrachtet. Jeder neue Patient wird zu eine der Behandlungsruppen zugeteilt und zwar so, dass die Unausgewogenheit in den Untergruppen minimiert wird. So kann auch in kleinen Studien die Vergleichbarkeit der Behandlungsgruppen gewährleistet werden.

2.3.3. Response-adaptive Randomisationsverfahren

Aus ethischen Gründen ist das Ziel von klinischen Studien nicht nur der Nachweis der Wirksamkeit der neuen Medikamente, sondern auch die bestmögliche Behandlung der teilnehmenden Patienten. Es ist schwer den Kompromiss zwischen diesen beiden Zielen zu finden. Eine mögliche Lösung dafür wäre das Benutzen der response-adaptiven Randomisation während der Studie. Die Wahrscheinlichkeit der Zuordnung der neuen Patienten zu einer der Therapiegruppen wird nach jeder Belegung oder nach jeder Rückantwort verändert. So werden schon während der Studie die Erfahrungen gesammelt, damit die später eingetroffenen Patienten mehr Chancen haben, bessere Behandlung zu bekommen. Das ist der wichtigste Unterschied zu den klassischen Randomisationsverfahren, in denen die Randomisation entweder noch vor dem Anfang der Studie vorbereitet wird oder von bestimmten Faktoren abhängt, die nichts mit der Qualität der Behandlung zu tun haben.

Um bei den klassischen Randomisationsverfahren eine möglichst große Power zu erhalten, müssen am Anfang der Studie bestimmte Größenverhältnisse der Therapiegruppen eingehalten werden. Bei response-adaptiven Randomisationsverfahren verhält sich die Power anders. Wenn die Anzahl der Patienten in den Gruppen ausgeglichen ist, dann müssen auch die Rückantworten nach Durchführung der Therapien ausgeglichen sein, damit die Power groß genug wird. Ansonsten wird die Power bei unausgeglichener Verteilung der Patienten maximiert [12].

Die response-adaptive Randomisationsverfahren können in zwei Klassen unterteilt werden [13]:

1. Zielorientierte response-adaptive Verfahren. Sie werden eingesetzt, wenn das Gleichgewicht zwischen den Behandlungsgruppen unerwünscht ist und die gewünschte Allokation unbekannt ist [14].(z.B. doubly adaptive biased coin design)
2. Designorientierte response-adaptive Verfahren. In diesen Verfahren werden intuitive Algorithmen benutzt, um die Zuordnungswahrscheinlichkeiten sequentiell zu verändern. (z.B. randomisierte play-the-winner Regel).

Die response-adaptiven Randomisationsverfahren bringen auch Probleme mit sich, wie z.B.:

- Die RPW[9] Regel (-> Seite 21) kann nur mit dichotomen Antworten umgehen, aber in der Realität haben die Therapien oft kontinuierliche oder polychotome Antworten (z.B. Blutdruck als Ergebnis) [15].
- Problematisch ist es auch bei den Studien mit verspäteten Antworten. Es kommt vor, dass nur wenige Antworten bekannt sind, während die meisten Patienten schon die Behandlung bekommen. Daher eignen sich diese Verfahren eher bei Studien bei denen das Ergebnis frühzeitig vorliegt. Es wurden die Eigenschaften der DBCD[10] (-> Seite 23) bei verspäteten Antworten untersucht, und bewiesen, dass dieses Verfahren in dieser Situation Vorteile gegenüber anderen response-adaptiven Verfahren hat [13].

2.3.4. Beschreibung der response-adaptiven Verfahren

Play-the-winner (PW) Regel

Dieses Verfahren kommt zum Einsatz, wenn es um zwei Vergleichstherapien (Therapie A und Therapie B) mit dichotomer Rückantwort (Erfolg oder Misserfolg) geht. Die Rückantwort soll dabei nur von der Therapie abhängen und nicht von anderen Faktoren [16]. Das Verfahren wurde 1969 von Zelen vorgeschlagen [17]. Für die Realisierung der PW Regel wird das Urnenmodell verwendet. Die Urne ist am Anfang leer. Deswegen erfolgt die Zuteilung des ersten Patienten nach dem Prinzip des Wurfes einer idealen Münze. Jedes Mal, wenn eine positive Rückantwort der Therapie A vorliegt, wird eine mit „A“ markierte Kugel in die Urne hineingelegt, und im Falle des Misserfolgs wird die Kugel mit „B“ markiert. Andernfalls, wenn die Therapie B erfolgreich abgeschlossen wird, wird eine Kugel markiert mit „B“ in die Urne hineingelegt, ansonsten wird sie mit „A“ markiert. Wenn ein neuer Patient in die Studie eintritt, wird zufällig eine Kugel aus der Urne gezogen. Die Ziehung erfolgt ohne Zurücklegen. In dem Fall, dass keine Kugeln mehr vorhanden sind, wird die Zuordnung genauso wie zu Beginn mit einer idealer Münze entschieden. In der Realität dauert es eine Weile bis die ersten Rückantworten bekannt werden. Da die Urne auch solange leer bleibt, ist diese Regel in den meisten Fällen nicht anwendbar.

[9] Randomisierte Play-the-winner

[10] Doubly biased coin design.

Modifizierte play-the-winner (MPW) Regel

Die MPW Regel ist eine Weiterentwicklung der PW Regel, mit der Bedingung, dass die Antwort nach der Behandlung sofort zurückkommt. Somit wird nach jedem Erfolg dieselbe Therapie zu dem nächsten Patienten zugeordnet und nach jedem Misserfolg wird die Therapie gewechselt. Mit so einem Verfahren bekommen mehr Patienten die bessere Behandlung, aber es darf keine Verzögerung bei den Rückantworten geben. Die MPW Regel ist aber deterministisch und damit Vorhersehbar. Bei der Bekanntgabe des Erfolges oder Misserfolges einer Behandlung, ist auch bekannt, welche Therapie der nächste Patient bekommen wird. Deswegen ist die Wahrscheinlichkeit, dass eine Auswahlverzerrung auftritt, sehr hoch [17].

Randomisierte play-the-winner (RPW) Regel

Es wird also weiter nach einem Verfahren gesucht, das im Gegensatz zur MPW Regel, nicht so deterministisch und nicht so anfällig für Verzerrungen ist. 1978 stellten Wei & Durham die RPW Regel vor [17]. Diese ist wie folgt aufgebaut:

Angenommen:

- Die Rückantwort der Behandlung ist dichotom (Erfolg oder Misserfolg)
- Die Wahrscheinlichkeit, dass die Behandlung i Erfolg haben wird ist p_i, mit $0<p_i<1$ und i=A, B.

Am Anfang liegen in der Urne u Bälle markiert mit „A“ und genauso viele Bälle markiert mit „B“. Bei dem Eintritt eines neuen Patienten in die Studie, wird zufällig ein Ball aus der Urne mit Zurücklegen gezogen. Der Patient wird die Therapie A bekommen, falls der gezogene Ball mit „A“ markiert ist, ansonsten die Therapie B. Nachdem eine Rückantwort von einer der Therapien bekannt ist, wird die Urne aktualisiert:

- Es liegt die Rückantwort von Therapie A vor:

  Im Fall des Erfolgs, werden zusätzliche β Bälle vom Typ „A“ und zusätzliche α Bälle vom Typ „B“ in die Urne hineingelegt.

  Im Fall des Misserfolgs, werden zusätzliche α Bälle vom Typ „A“ und zusätzliche β Bälle vom Typ „B“ in die Urne hineingelegt.

- Es liegt die Rückantwort von Therapie B vor:

  Im Fall des Erfolgs, werden zusätzliche β Bälle vom Typ „B“ und zusätzliche α Bälle vom Typ „A“ in die Urne hineingelegt.

  Im Fall des Misserfolgs, werden zusätzliche α Bälle vom Typ „B“ und zusätzliche β Bälle vom Typ „A“ in die Urne hineingelegt.

- Dabei gilt: $\beta \geq \alpha \geq 0$.

    - je größer β/α ist, desto mehr Patienten bekommen tendenziell die bessere Therapie.
    - je kleiner β/α ist, desto größer ist die Wahrscheinlichkeit einer zufälligen Zuordnung [17].

Also nach jeder Rückantwort werden genau α+β Bälle hinzugelegt. Diese Regel wird als RPW(u, α, β) bezeichnet und ist auch anwendbar, wenn die Ergebnisse nicht sofort vorhanden sind.

Der Unterschied zwischen PW Regel und RPW(0, 0, 1) ist nur, dass im ersten Fall die Bälle ohne Zurücklegen gezogen werden.

Für die großen Studien ist RPW(u, 0, β) genauso gut wie die MPW Regel [17].

Stratifizierte und randomiesierte play-the-Winner (SRPW) Regel

SRPW Regel ist noch ein weiteres Randomisationsverfahren, das aus dem PW Regel entstanden ist. Die wichtigsten Vorteile der SRPW Regel sind [18]:

- Asymmetrische Zuweisung der Patienten zur besseren Therapie. Das Verfahren ist auch bei Studien mit verzögerten Rückantworten anwendbar (was auch schon mit RPW Regel möglich war)
- Ist anwendbar bei klinischen Studien mit mehr als zwei Vergleichstherapien
- Ermöglicht Vergleich der Therapiegruppen bezüglich den wichtigen prognostischen Faktoren

Bei der Bezeichnung SRPW(μ, α, β, t, s) ist:

t = Anzahl der Therapien;

s = Anzahl der Stratifikationsmerkmale;

μ, α, β = Anzahl der Bälle,

mit $\mu \geq 0$,

α und β sind Vielfaches von $(t-1)$,

und $0 \leq \alpha(t-1) \leq \beta$ [18].

Der Ablauf des Algorithmus ist folgendermaßen:

- Es werden s Stratifikationsmerkmale definiert
- Für jedes Merkmal wird eine Urne vorbereitet. In jede Urne werden jeweils μ Bälle markiert mit i ($i = 1, 2, \ldots, t$) für jede der t Therapiemöglichkeiten hineingelegt.
- Ein neuer Patient, der über einen bestimmten Stratifikationsmerkmal k ($k=0, 1, ..., s$) verfügt, wird in die Studie aufgenommen. Aus der k-ten Urne wird ein Ball mit Zurücklegen gezogen. Der Patient bekommt die Therapie i, die dem Typ des Balles entspricht. Falls die Urne leer ist, wird eine Zufallszahl zwischen 0 und 1 generiert. Der Patient wird zu der Therapie i zugeordnet, falls die generierte Zahl kleiner als i/t und größer oder gleich als $(i-1)/t$ ist.
- Nachdem ein Ergebnis einer durchgeführten Behandlung vorliegt, wird die Anzahl der Bälle in der Urne geändert. Angenommen der schon behandelte Patient gehörte zu der Gruppe mit dem Stratifikationsmerkmal k. Dann werden im Fall der erfolgreich abgeschlossenen Behandlung zusätzliche β Bälle markiert mit i und zusätzliche α Bälle von anderen Typen (jeweils $\alpha/(t-1)$ von jedem Typ) entsprechend in die k-te Urne hineingelegt. Im Fall des Misserfolgs werden zusätzliche α Bälle markiert mit i und zusätzliche β Bälle von anderen Typen (jeweils $\beta/(t-1)$ von jedem Typ) entsprechend in die k-te Urne hineingelegt. Die RPW Regel ist ein Spezialfall der SRPW (μ, α, β, t, s) mit $t=2$ und $s=1$.

Doubly adaptive biased coin design (DBCD)

Alle bis jetzt vorgestellten Randomisationsverfahren gehören zu der Familie der designorientierten response-adaptiven Verfahren. Die Algorithmen von diesen Verfahren sind typischerweise intuitiv [19]. Die klinischen Studien stellen aber normalerweise sehr komplexe Experimente auf Menschen (oft kranke Menschen) mit mehreren Zielen dar. Manche von diesen Zielen sind [20]:

- Power maximieren, um den klinisch relevanten Unterschied zwischen den Behandlungen festzustellen;
- Qualität der Behandlung jedes Patienten maximieren;
- Kosten minimieren.

Die optimale Allokation bei den zielorientierten response-adaptiven Verfahren wird mit Hilfe eines speziellen Kriteriums durchgeführt. Dieses Kriterium wird basierend auf dem Modell, das die Antworten der Behandlungen der Studienteilnehmer beschreibt, optimiert. Das DBCD wurde zuerst von Eisele & Woodroofe [21] vorgestellt (später von Hu & Zhang [14]).Das DBCD hat einige wichtige Vorteile, gegenüber anderen Randomisationsverfahren [19]:

- Im Vergleich zu den anderen Randomisationsverfahren, mit der gleichen Einschränkung der Größenverhältnisse der Therapiegruppen, verursacht das DBCD weniger asymptotischer Varianz. Das bedeutet, dass die Power größer ist und der Therapievergleich aussagekräftiger.

- Das DBCD kann sowohl bei schnellen, ununterbrochenen als auch bei verzögerten, diskreten Rückantworten angewendet werden.

Angenommen, zwei Therapien sollen verglichen werden:

- $X_m = (X_{m,1}, X_{m,2})$ ist die Therapie, die der Patient m bekommt.
- $\xi_m = (\xi_{m,1}, \xi_{m,2})$ ist die Rückantwort nach der durchgeführten Therapie.
- $N_n = (N_{n,1}, N_{n,2})$ mit $N_{n,k} = \sum_{j=1}^{n} X_{j,k}$ ist die Anzahl der ersten n Patienten, die zu der Therapie k=1, 2 beigetreten sind. Dabei $N_{n,1}+N_{n,2}=n$.
- θ_1 und θ_2 sind die zugehörigen Parameter, jeweils zu Therapie 1 und 2. Basierend auf den ersten n Beobachtungen ist $\hat{\theta}_{n,1}$ die Schätzfunktion von θ_1 und $\hat{\theta}_{n,2}$ die Schätzfunktion von θ_2.
- Die Funktion g(x, y) von [1,0]x[1,0] bis [0,1] beschreibt bei der i-ten Allokation, wie nahe $N_1(i-1)/(i-1)$ zu der aktuellen Schätzung von $p(\theta)$ ist [19].

Am Anfang wurden $n_0 \geq 2$ Patienten zu den beiden Therapiegruppen 1 und 2 zugeordnet. Bei dem Schritt (m+1)(mit $m \geq 2n_0$) wird der (m+1)-er Patient die

Therapie 1 mit der Wahrscheinlichkeit $g(N_{m,1}/m, \hat{p}_m)$ bekommen. Dabei ist g die Allokationsfunktion und $\hat{p}_m = p(\hat{\theta}_{m,1}, \hat{\theta}_{m,2})$

Hu & Zhang (2004) haben folgende Funktion für g(x, y) vorgeschlagen [14]:

$$g(x, y) = \frac{y(y/x)^{\gamma}}{y(y/x)^{\gamma}+(1-y)((1-y)/(1-x))^{\gamma}}, \text{ mit } \gamma \geq 0,$$

$$g(0, y) = 1,$$

$$g(1, y) = 0.$$

Mit der Anwendung dieser Regel wird die bessere Therapie bevorzugt und sie bekommen mehr Patienten.

2.3.5. Die Wahl des Verfahrens

Die Implementierung der response-adaptiven Randomisationsverfahren in der Anwendung RANDI2 ist wünschenswert, um auch Studiendesigns mit speziellen ethischen Anforderungen zu unterstützen. Ziel dieser Arbeit ist eines der Verfahren, genauer die SRPW Regel, zu implementieren. Dieses Verfahren wurde von N. Waskowzow mit D. Schrimpf ausgewählt, da es viele Vorteile mit sich mitbringt. So wie:

- Möglichkeit die Studien mit mehr als zwei Therapien zu randomisieren,
- Möglichkeit den Studienteilnehmern nach verschiedenen Stratifikationsmerkmale aufzuteilen,
- Hohe Testbarkeit,
- Und die intuitive Umsetzung.

3. RANDI2: Ist-Analyse

3.1. Funktionalitäten

In der aktuellen Version 0.7 von RANDI2 sind folgende Verfahren zur Randomisation von klinischen Studien realisiert: Vollständige Randomisation, Schiefer Münzwurf, Trunkierte Randomisation, Blockrandomisation (mit fester und variabler Blocklänge), Minimisation und Wei's Urnenmodell. Es besteht auch die Möglichkeit, bei geeigneten Verfahren, die Studienteilnehmern nach verschiedenen Merkmale zu stratifizieren. Die Merkmale können dichotom (z.B. Geschlecht), ordinal (z.B. Tumorstatus), im Datumsformat (z.B. Geburtsdatum) oder im Form des freien Textes sein[11].

Jeder Benutzer kann zwischen zwei Sprachen auswählen: Englisch oder Deutsch.

In der aktueller Version gibt es fünf verschiedene vorkonfigurierte Benutzerrollen. D.Schrimpf beschreibt die Rollen in seiner Diplomarbeit [10] wie folgt:

- „Administrator, diese Rolle beinhaltet administrative Aufgaben, darunter zählen das Anlegen und Ändern von Studienzentren und beliebigen Personendaten. Ein Administrator ist jedoch nicht berechtigt, Studien anzulegen oder Patienten zu randomisieren.
- Studienleiter, ein Benutzer mit dieser Rolle ist berechtigt Studien anzulegen und diese unter bestimmten Voraussetzungen zu ändern. Es ist ihm auch möglich, den Verlauf der Studie zu betrachten. Weiterhin ist es mit dieser Rolle erlaubt, Personen im eigenen Studienzentrum anzulegen. Die Randomisation von Patienten ist mit dieser Rolle jedoch nicht möglich.
- Prüfarzt, dieser Benutzer hat das Recht Patienten für die Studien in seinem Zentrum zu randomisieren und er kann seine Patienten mit dem Randomisationsergebnis betrachten.
- Statistiker, diese Rolle dient der Kontrolle und Auswertung von klinischen Studien. Daher kann der Statistiker den Verlauf der Studie, angebotene Auswertungen und die Randomisationsergebnisse aufrufen.

[11] www.randi2.org

- Monitor, die Rolle des Monitors überwacht die Studiendurchführung und die Qualität der Daten. Daher ist es dieser Rolle erlaubt, die Studiendurchführung und die randomisierten Patienten zu prüfen.“

Einem Benutzer können gleichzeitig mehrere Rollen zugeordnet werden, damit er verschiedene Aufgaben erledigen kann. Die beiden nächsten Diagramme zeigen die Funktionen, die ein Benutzer in verschiedenen Rollen erledigen kann.

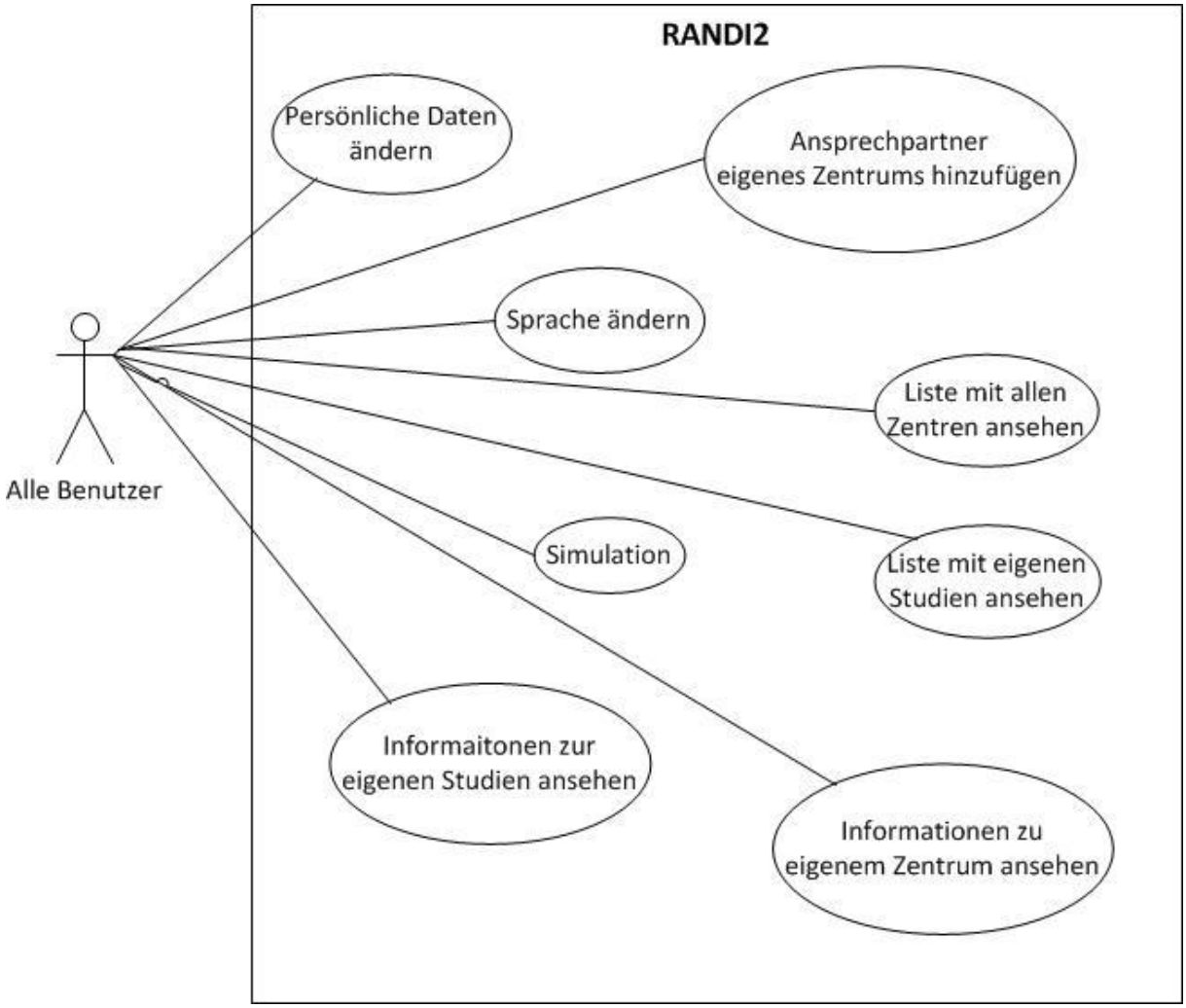

Abbildung 3.1. 1: Anwendungsfalldiagramm: Funktionen, die ein Benutzer in jeder Rolle erledigen kann.

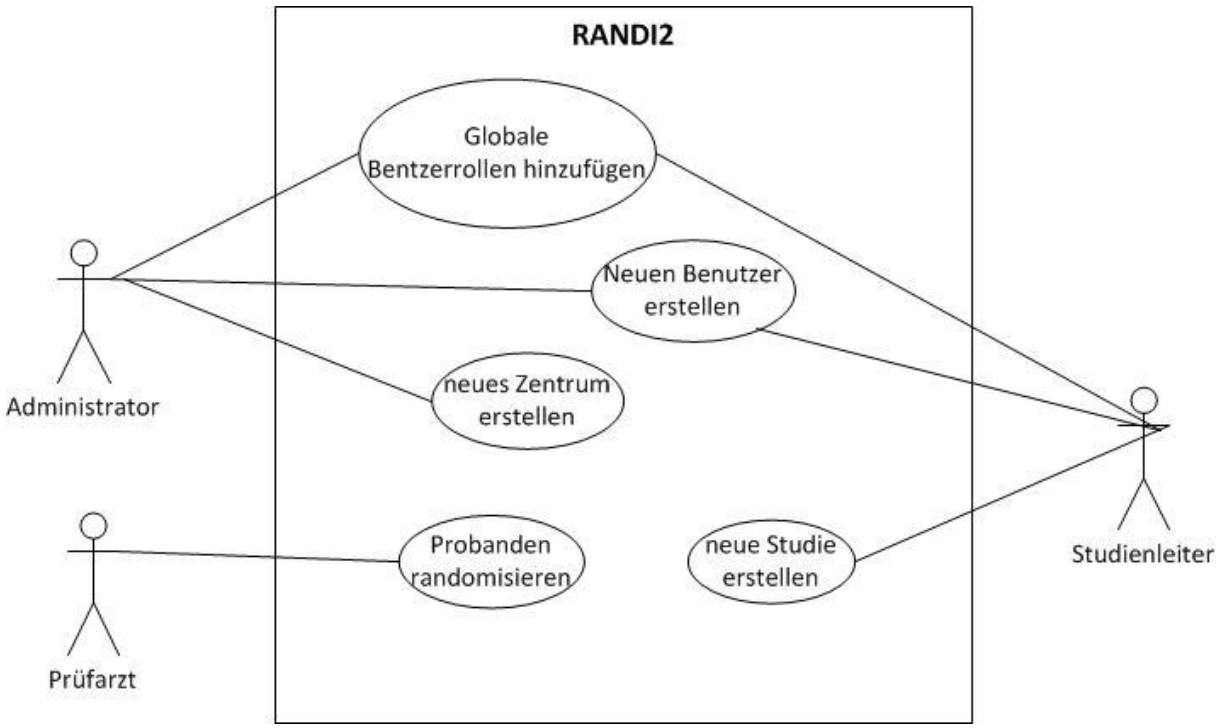

Abbildung 3.1. 2: Anwendungsfalldiagramm: Zusätzliche Funktionen der Benutzer

Um den Patienten randomisieren zu können, muss zuerst vom Studienleiter eine neue Studie erstellt werden. Das erfolgt momentan in fünf Schritte:

- Schritt 1: Hier werden der Name und die Abkürzung für die neue Studie definiert. Zudem hat der Benutzer die Möglichkeit eine Beschreibung hinzuzufügen. Außerdem werden die Zeitperiode für die Durchführung der Studie, das leitende Zentrum, sowie der Studienleiter festgelegt.
- Schritt 2: Im zweiten Schritt werden an der Studie teilnehmenden Zentren festgelegt.
- Schritt 3: Hier werden Anzahl, Namen, Beschreibungen und die geplante Größe der Behandlungsarme[12] angegeben.
- Schritt 4: In diesem Schritt hat der Benutzer die Möglichkeit die gewünschten Patienteneigenschaften einzutragen.
- Schritt 5: Im letzten Schritt wählt der Benutzer als erstes die Randomisationsart, in Abhängigkeit, welches Verfahren ausgewählt wurde. In Abhängigkeit, welche Art ausgewählt wurde, erscheinen entweder weiteren Eingabemasken (mit den Eigenschaften für die jeweilige Randomisation) oder alle bisher eingegebenen Daten können gespeichert werden. Die Vollständige

[12] Ein Behandlungsarm ist eine Therapiemöglichkeit und Größe des Behandlungsarmes ist die Anzahl der Patienten, die diese Therapie bekommen

Randomisation, Schiefer Münzwurf und die Trunkierte Randomisation benötigen keine weiteren Eingaben. Bei der Blockrandomisation, Minimisation und Wei's Urnenmodell ist die Eingabe weiteren Konfigurationsdaten nötig. Die letzten drei Verfahren haben auch die Möglichkeit zur Stratifikation (Abbildung 3.1.3 – Beispiel der Eingabemaske für die Blockrandomisation), dafür werden aus den Patienteneigenschaften, die in Schritt 4 definiert wurden, die Stratifikationsmerkmale bestimmt. Falls eine Stratifikation gewählt wurde, wird zusätzlich nachgefragt, ob die Stratifikation nach den Prüfzentren gewünscht ist.

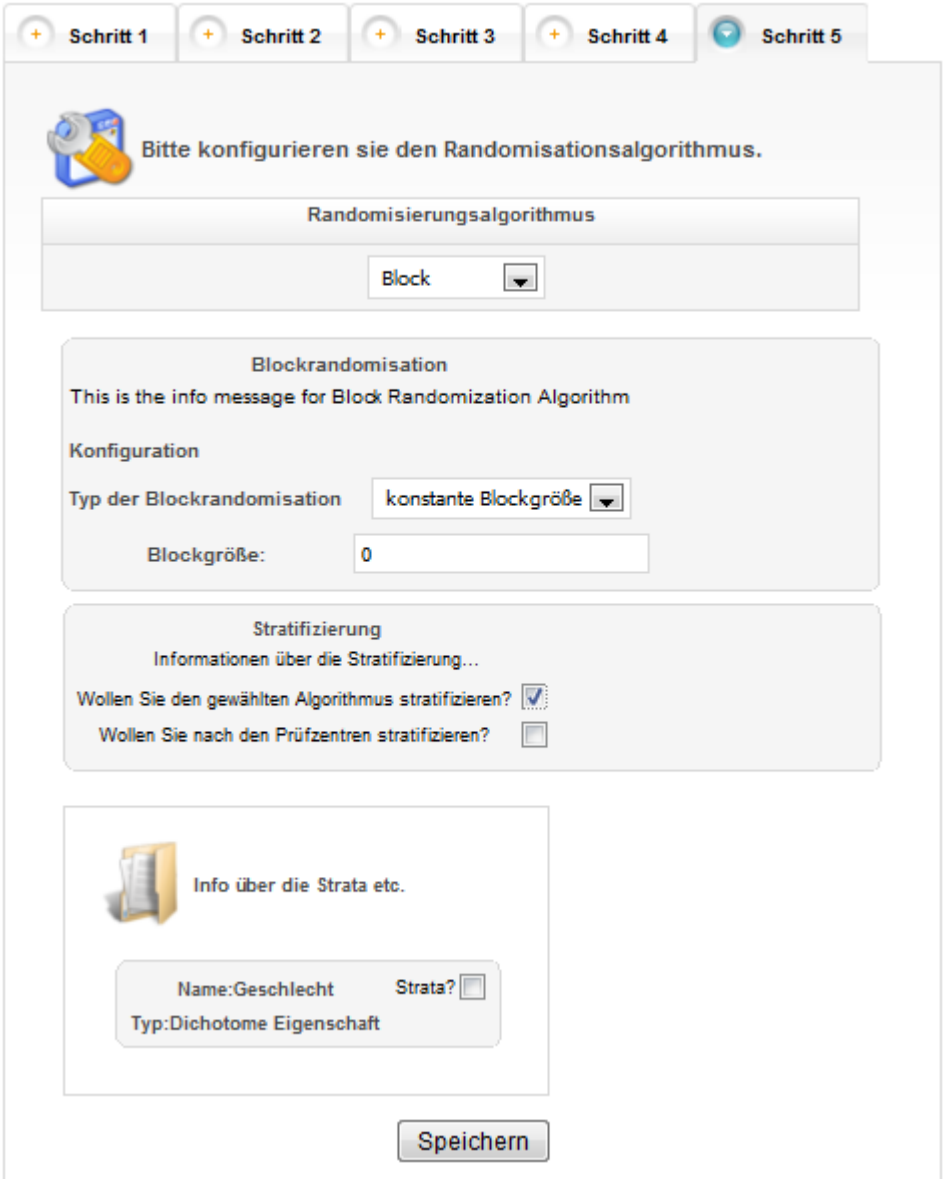

Abbildung 3.1. 3: Eingabemaske für die Blockrandomisation

Nachdem eine Studie angelegt wurde, kann der Prüfarzt neue Patienten hinzufügen (Abbildung 3.1.4 – Funktionen, die der Prüfarzt erledigen kann), dafür muss er zuerst

diese Studie auswählen und kann anschließend den Patienten randomisieren. Bei der Aufnahme eines neuen Patienten in die Studie, werden für die Randomisation wichtige Merkmale von ihm abgefragt (z.B. Patienteneigenschaften, Abbildung 3.1.5), diese Eingabemaske ist spezifisch für jede Studie. Da es noch keiner der response-adaptiven Verfahren realisiert ist, wird die Rückantwort nach der Behandlung nicht erfasst.

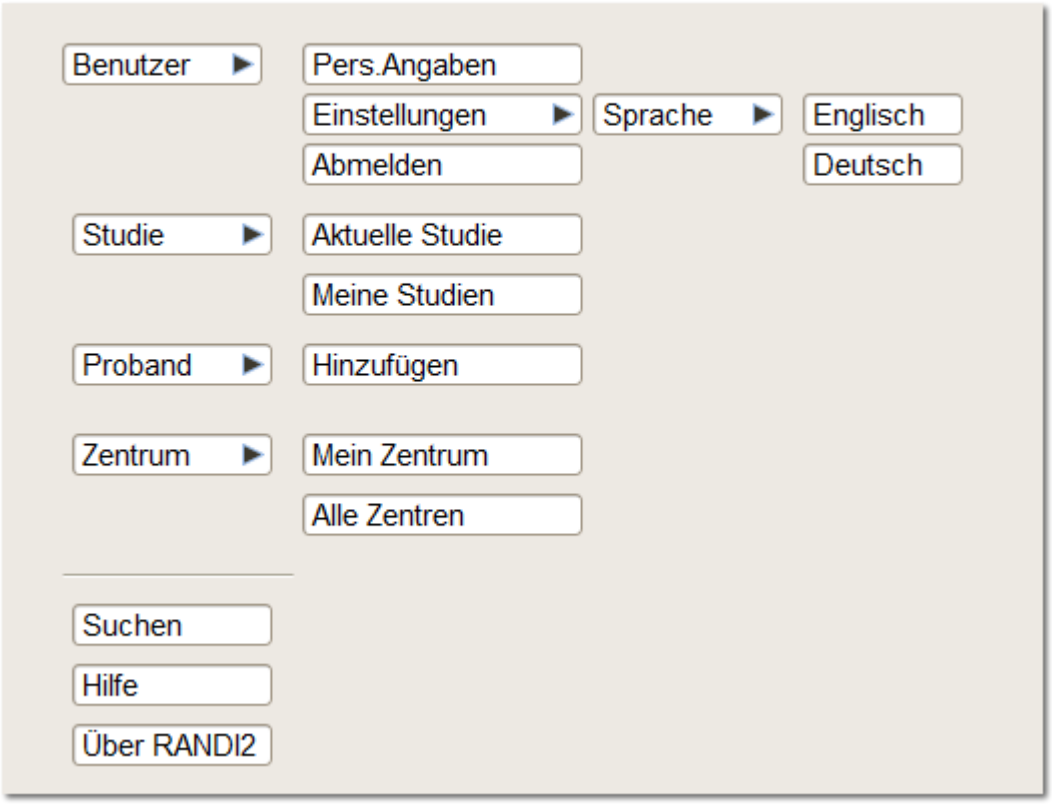

Abbildung 3.1. 4: Anzeigemenü des Benutzers „Prüfarzt“

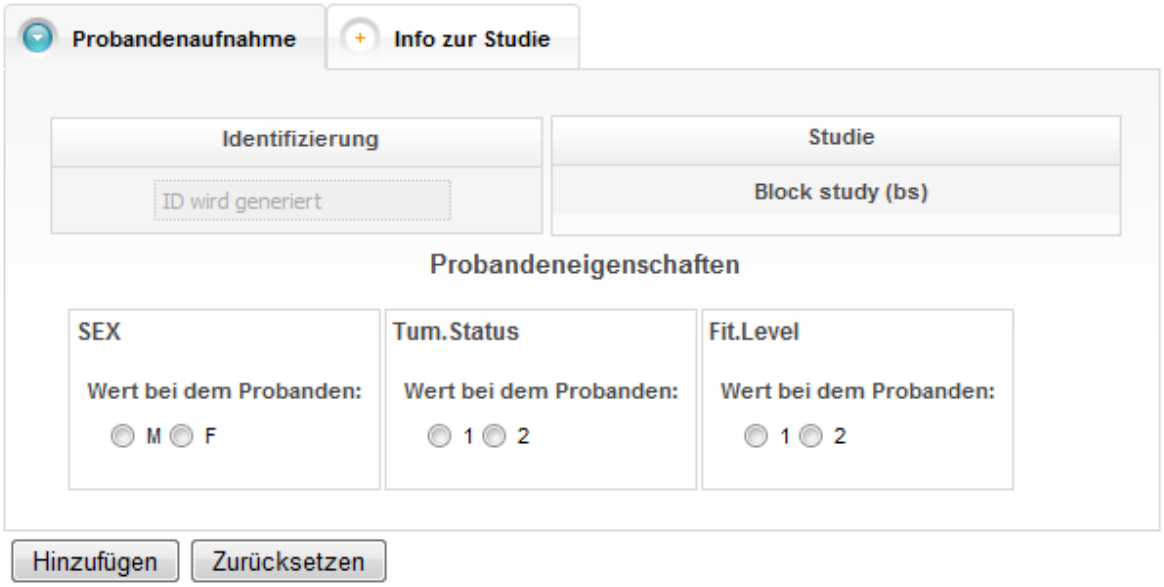

Abbildung 3.1. 5: Randomisation eines neuen Patienten

Jeder Zeit kann ein Benutzer alle Informationen zu seinen Studien ansehen. In Abhängigkeit, welche Rolle der Benutzer hat, können die Informationen abweichen. Die Benutzer mit den Rollen „Studienleiter“, „Monitor“ oder „Statistiker“ sehen folgende Informationen zu ihren Studien:

- Allgemeine Informationen: Name, Abkürzung, Beschreibung, Status, Start- und Enddatum, leitendes Zentrum und Studienleiter der Studie.
- Studienzentren: Namen und Adressen der teilnehmenden Studienzentren.
- Algorithmuseigenschaften: Typ des Algorithmus und seine Konfigurationseinstellungen.
- Randomisierungsdaten: Informationen über Probandenanzahl und ihre Verteilung (je Arm und je Zentrum) sowohl tabellarisch als auch grafisch dargestellt und das Diagramm zur Rekrutierung der Probanden über die Zeit. Der Benutzer kann zwischen unterschiedlichen Typen der Diagramme auswählen und auch die Daten als .csv Datei exportieren.
- Audit: Verlauf der Studie.

Der Administrator kann nur die allgemeine Informationen und Studienzentren sehen. Die Ansicht einer Studie bei dem Prüfarzt ist ähnlich wie die des Administrators, außer dass dem Prüfarzt zusätzlich die Information über seine Patienten angezeigt wird.

3.2. Analyse des Domain-Modelles

Die Klasse, die einen Patienten darstellt, befindet sich in dem Paket *model.* Die Abbildung 3.2.1 stellt dieses Paket mit seinen Unterpaketen dar.
Zu dem Inhalt des Paketes *model* gehören die Fachklassen der Anwendung. Folgende Klassen sind direkt dem Paket zugeordnet: *Login, Person, Role, SubjectProperty, TreatmentArm, Trial, TrialSite, TrialSubject,* genauso wie die abstrakte Klasse *AbstractDomainObject.*
In den Klassen des Paketes *criteria* mit seinem Unterpaket *constraint* werden die Patienteneigenschaften definiert und verwaltet.
In dem Paket *randomization* befinden sich die Klassen die für die Konfiguration der Ransomisationsalgorithmen und für die Verwaltung der Temporären Daten, die während der Randomisation entstehen, zuständig sind.

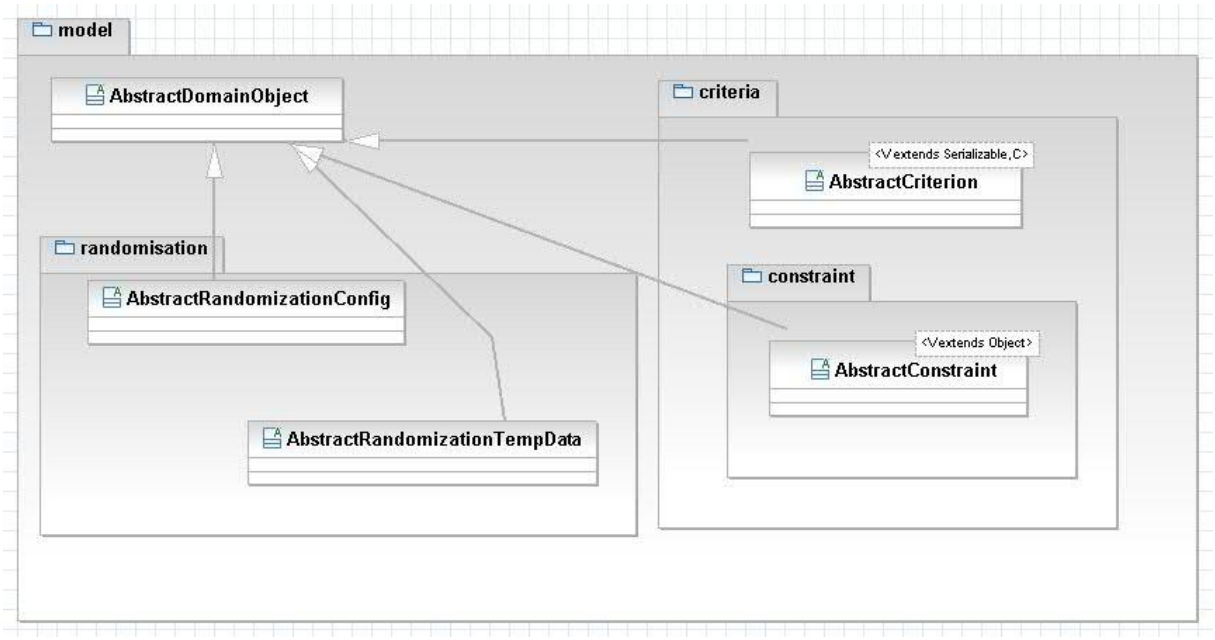

Abbildung 3.2. 1: Paketdiagramm: model

Auf der Abbildung 3.2.2 sind die Klassen, die einen Patienten mit allen seinen Eigenschaften darstellen, abgebildet. Die Klasse *TrialSubject* stellt einen Patienten dar. Wie aus dem Diagramm zu sehen ist, kann ein Patient über verschiedene Eigenschaften (*properties)* verfügen. Diese Eigenschaften sind vom Typ *SubjectProperty* und können zum Beispiel die Stratifikationsmerkmale sein. Jede dieser Eigenschaften gehört zu einem der definierten Kriterien(dichotome/ordinale Eigenschaften, Datum oder freier Text). Über eine Eigenschaft, die das Ergebnis der Behandlung enthält, verfügt ein Patient noch nicht.

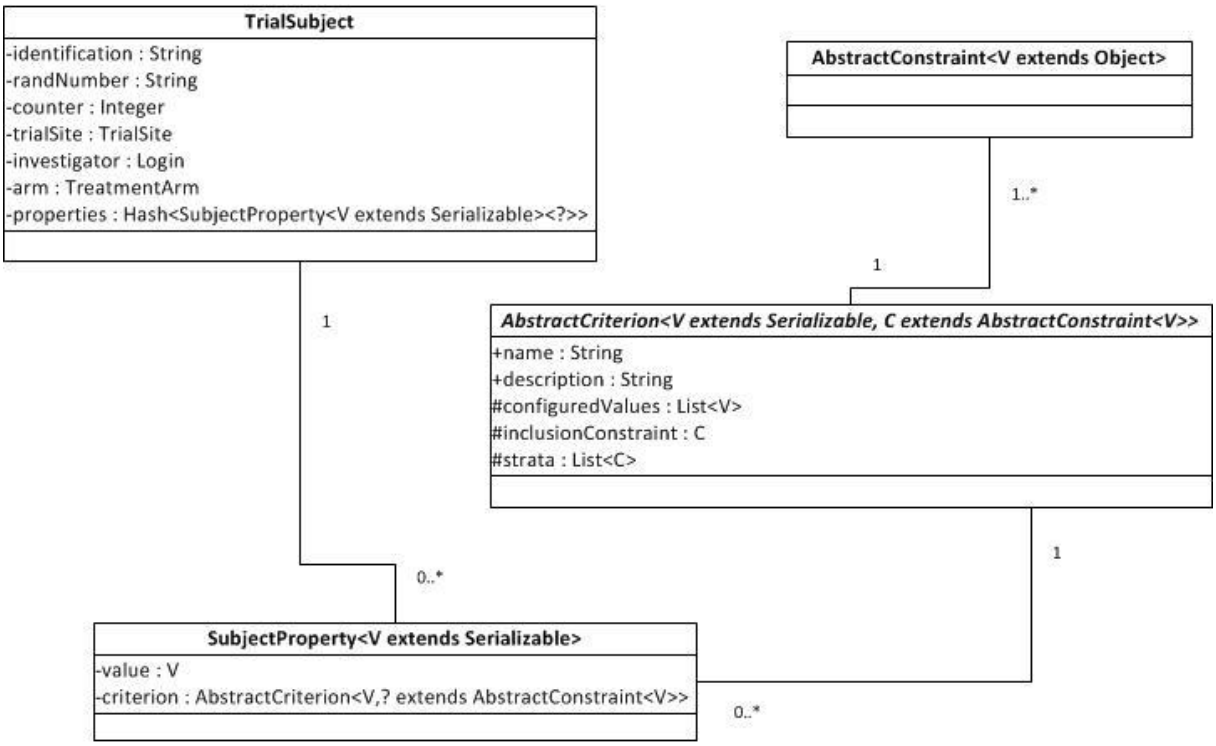

Abbildung 3.2. 2: Klassendiagramm: TrialSubject

Die Abbildung 3.2.3 zeigt die Realisierung einer Studie. Eine Studie(*Trial*) verfügt über mindestens zwei Behandlungen (*TreatmentArm*) zu denen jeweils die Patienten (*TrialSubject*) nach ausgewähltem Randomisationsalgorithmus (*AbstractRandomisationConfig*) randomisiert werden. Eine Studie ist auch mindestens zu einem Zentrum (*TrialSite*) zugewiesen.

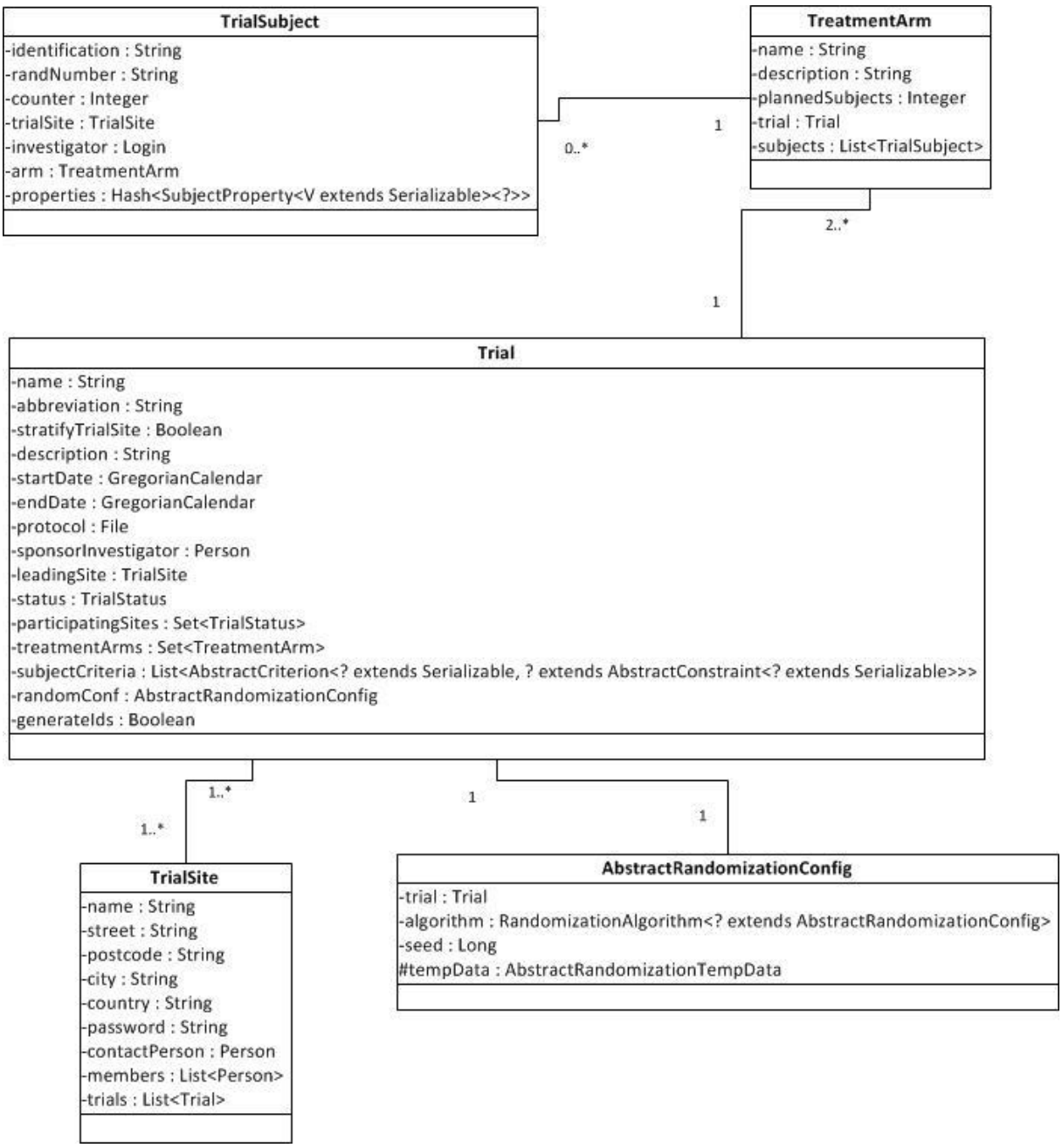

Abbildung 3.2. 3: Klassendiagramm: eine Studie mit den zugehörigen Eigenschaften.

Auf der Abbildung 3.2.4 ist die Realisierung des Urnenmodells nach Wei gezeigt (als Beispiel für die Realisierung der Randomisationsverfahren). Bei diesem Algorithmus wird für jedes Stratifikationsmerkmal eine Urne generiert, die am Anfang *"inizializeCountBalls"* Bälle enthält. Falls aus einer Urne ein Ball, der zum Beispiel die Therapie 1 darstellt, gezogen wird, wird in derselben Urne *"countReplacedBalls"* Bälle, die Therapie 2 darstellen, hineingelegt. Das Attribut *"urns"* der Klasse *UrnDesignTempData* beinhaltet schon generierte Urnen. Der Randomisationsalgorithmus ist in der Klasse *UrnDesign* realisiert.

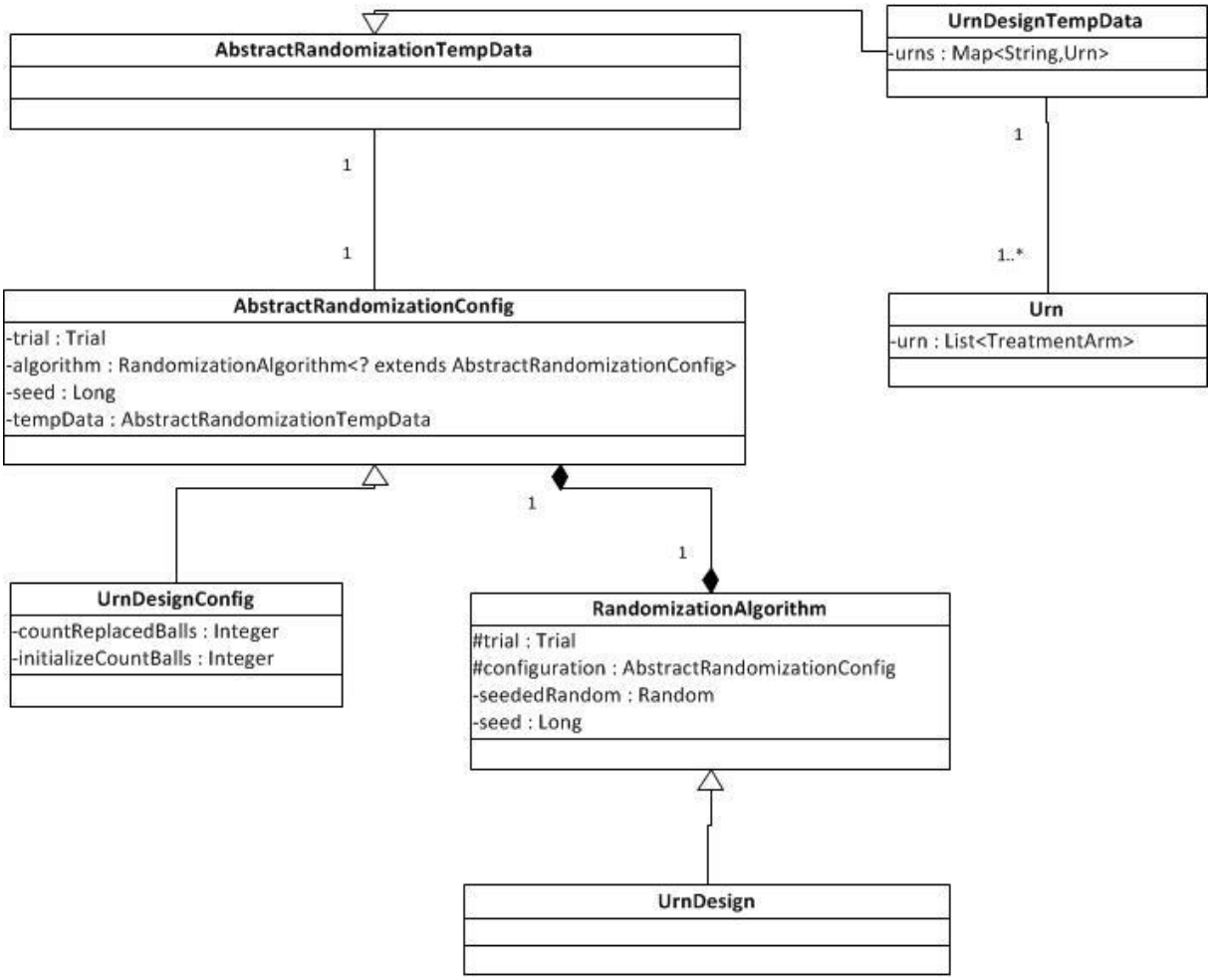

Abbildung 3.2. 4: Klassendiagramm: Urnenmodell nach Wei.

Wie die Randomisation allgemein realisiert ist, zeigt die Abbildung 3.2.5. Die Methode *3.3: doRandomize(TrialSubject subject, Random random)* ist abstrakt. In Abhängigkeit, welches Randomisationsverfahren bei der Studie ausgewählt wurde, wird diese Methode von der entsprechende Klasse (im Beispiel oben (Wei's Urnenmodell) - von der Klasse *UrnDesign*) implementiert und von dort aufgerufen.

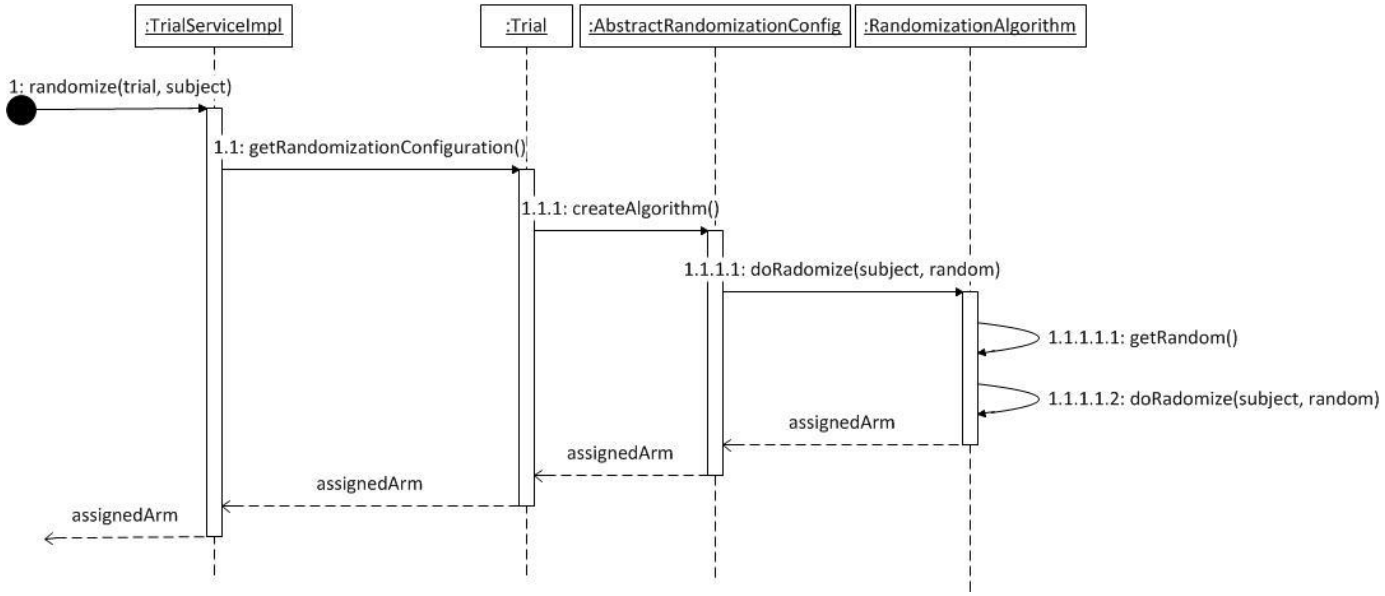

Abbildung 3.2. 5: Sequenzdiagramm: Randomisation eines Probanden

3.3. Erweiterungsmöglichkeiten

Für die Realisierung der response-adaptiven Randomisationsverfahren muss die Klasse *TrialSubject* die Möglichkeit anbieten, die Rückantworten zu speichern. Die Oberfläche, die dem Benutzer ermöglicht, das Ergebnis der Behandlung einzutragen, ist momentan nicht vorhanden. Aber durch bei RANDI2 verwendeten Technologien können beliebige Erweiterungen relativ einfach realisiert werden.

4. Konzeption

4.1. Oberflächenentwurf

Wie im Kapitel 3.1 beschrieben ist, hat nur der Prüfarzt die Rechte Studienteilnehmern zu randomisieren. Dafür kann er momentan, nachdem er die Studie zur Randomisation ausgewählt hat, im Menüpunkt „Proband“ den Untermenüpunkt „Hinzufügen“ auswählen. Da bei response-adaptiver Randomisation außer Patientendaten auch die Ergebnisse der Behandlungen notwendig sind, wird der Menüpunkt „Proband“ mit einem neuen Untermenüpunkt „Antwort hinzufügen“ erweitert. Und um Missverständnisse zu vermeiden, wird der bis jetzt existierender Untermenüpunkt „Hinzufügen“ in „Proband hinzufügen“ umbenannt (Abbildung 4.1.1)

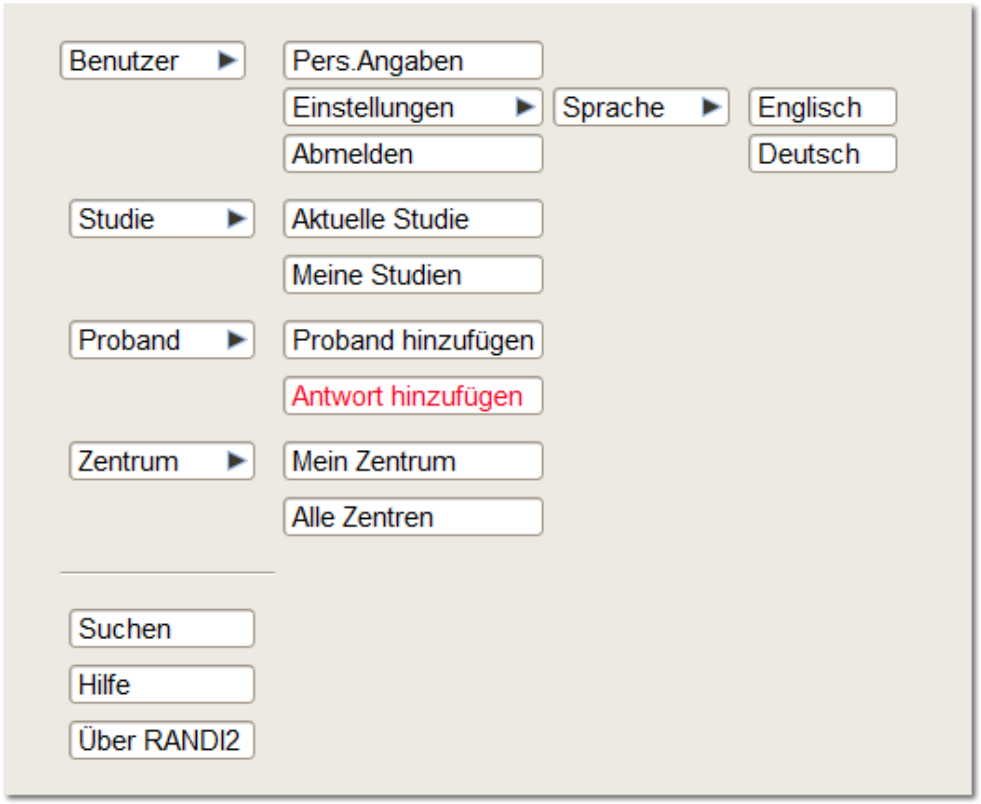

Abbildung 4.1. 1: Neues Anzeigemenü des Benutzers „Prüfarzt“

Nachdem der Prüfarzt den Menüpunkt "Antwort hinzufügen" ausgewählt hat, gelangt er zu der Eingabemaske für die Antwort (Abbildung 4.1.2). Als erstes sieht er nur ein Eingabefeld für die Id des Patienten. Nachdem der Patient mit der eingegebenen Id gefunden wurde, erscheinen seine Daten auf der Oberfläche (die Studie, bei der er teilnimmt, der Behandlungsarm und die Eigenschaften des Patienten). Darunter hat der Benutzer die Möglichkeit eine der bei Erstellen der Studie eingegebenen Antwortmöglichkeiten (Abbildung 4.1.3) auszuwählen und zu speichern.

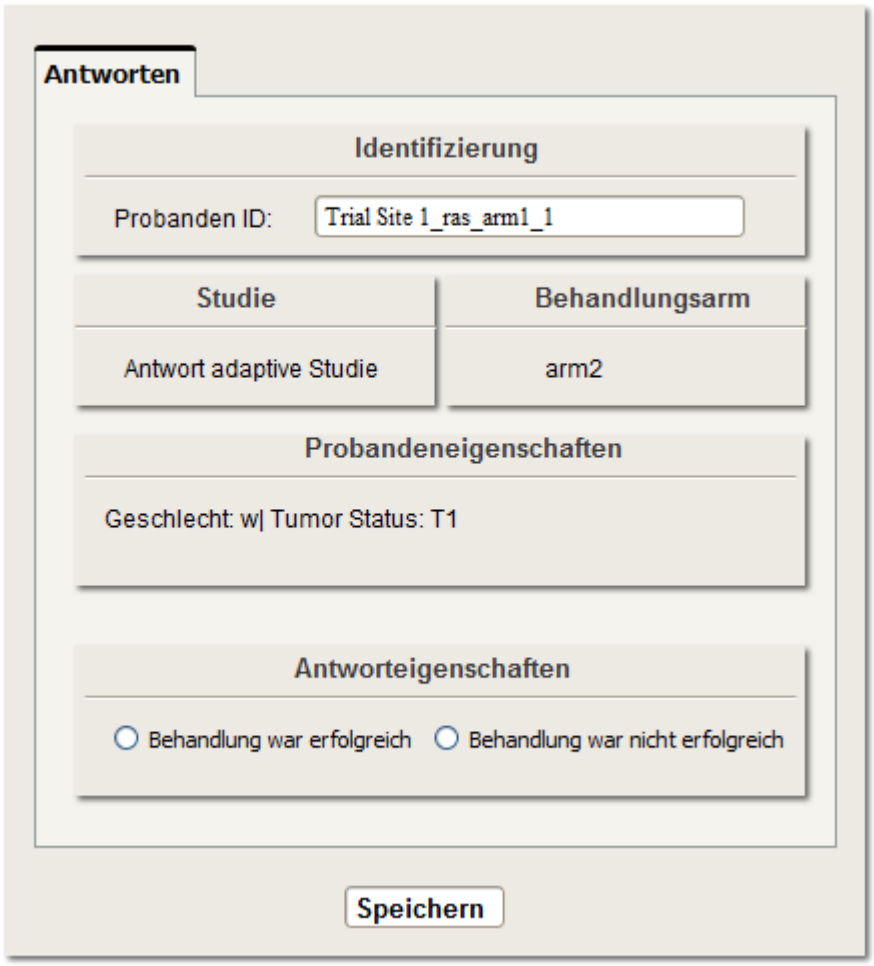

Abbildung 4.1. 2: Entwurf der Oberfläche zur Aufnahme der Antwort nach der Durchführung der Behandlung

Einer der fünf Schritte, die zur Erstellung einer Studie führen, ist die Auswahl des Randomisationsverfahrens und die Festlegung aller Konfigurationsdaten für das ausgewählte Verfahren, sowie der Stratifikationsmerkmale. Für die Realisierung der response-adaptiver Randomisation in RANDI2 soll im Schritt fünf eine neue Oberfläche angeboten werden, die die vollständige Konfiguration dieses Verfahrens ermöglicht. Die Abbildung 4.1.3 stellt einen Entwurf dieser Oberfläche dar. So wie bei

anderen schon vorhandenen Randomisationsverfahren, werden auch hier die speziellen Initialisierungsdaten abgefragt:

1. Anzahl Bälle am Anfang der Randomisation;
2. Anzahl der zurückgelegten Bälle:
    a) im Fall der erfolgreich abgeschlossener Behandlung;
    b) im Fall des Misserfolges

Die Möglichkeit der Stratifikation sowohl nach Patienten als auch nach Prüfzentren ändert sich nicht. Was hier im Gegensatz zu anderen Randomisationsalgorithmen neu ist, ist die Konfiguration der Antworteigenschaft. Momentan wird es nur die Möglichkeit geben, eine dichotome Antwort einzugeben.

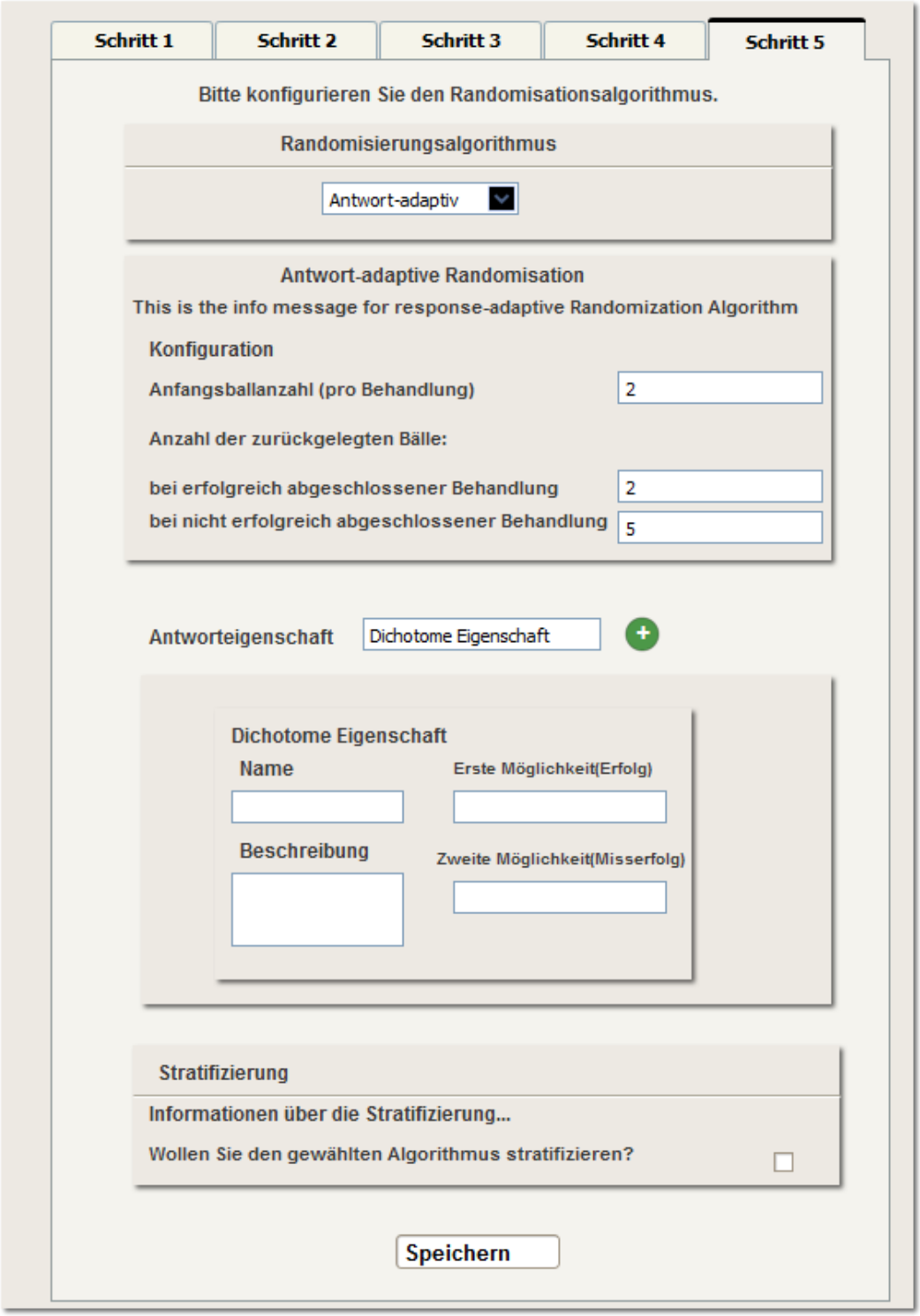

Abbildung 4.1. 3:Entwurf der Oberfläche zur Konfiguration der response- adaptiven Randomisation

Der Studienleiter, Monitor und Statistiker können bei der Informationen zur ihren Studien die Eigenschaften des Randomisationsalgorithmus sehen. Diese Oberfläche soll genauso wie die Oberfläche zur Konfiguration des Algorithmus für die response-

adaptive Randomisation mit der Möglichkeit Antworten einzugeben erweitert werden (Abbildung 4.1.4).

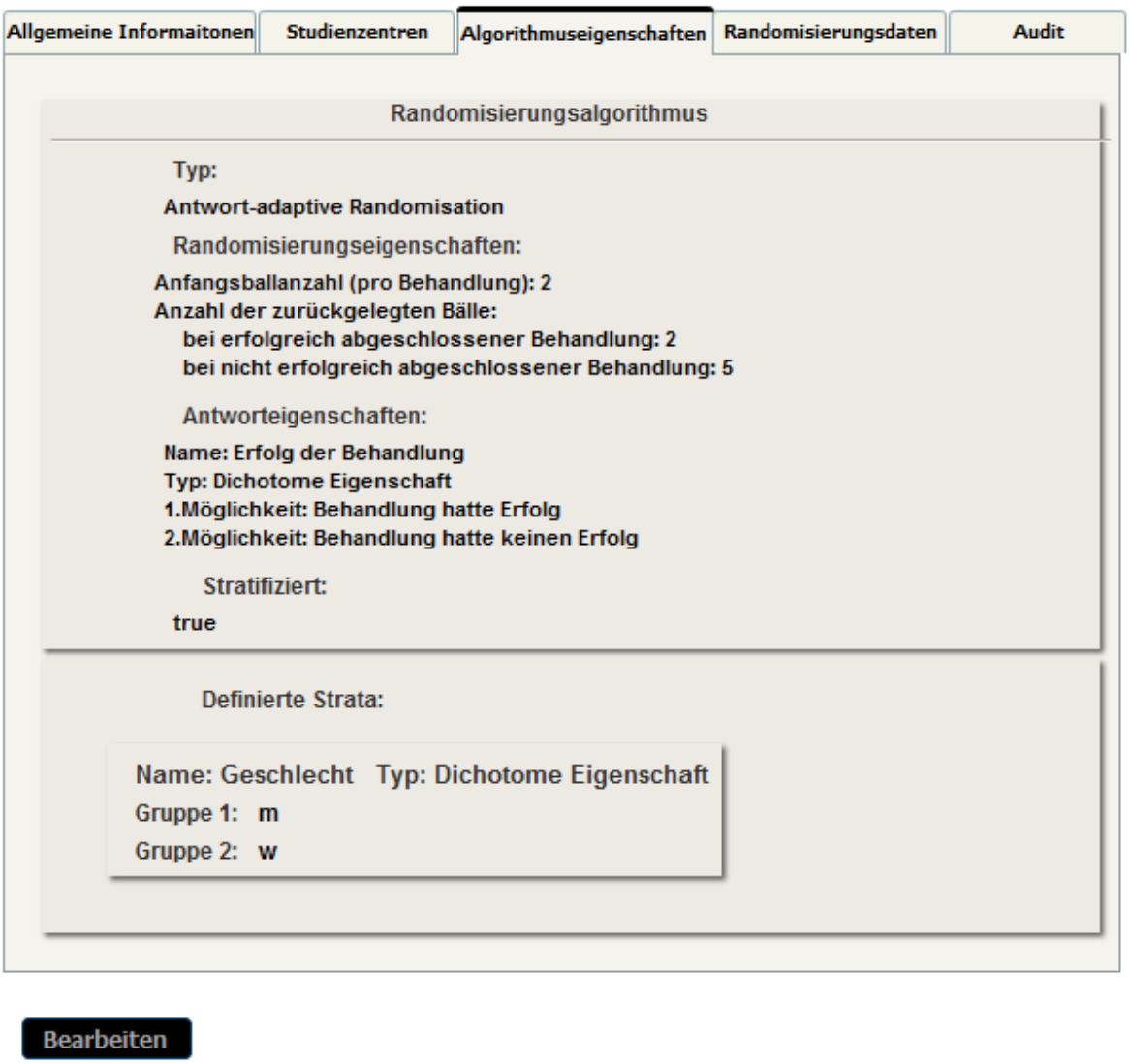

Abbildung 4.1. 4: Entwurf der Oberfläche für die Anzeige der Eigenschaften des Randomisationsalgorithmus einer Studie

4.2. Objektorientierte Analyse und Design

Um das neue Randomisationsverfahren zu implementieren, muss das Programm um vier neue Klassen erweitert werden:

- „ResponseAdaptiveRConfig“: hier werden die Konfigurationsdaten verwaltet.
- „ResponseAdaptiveRandomization“: hier wird der Algorithmus sowohl für die Randomisation eines neuen Patienten als auch für die Aufnahme einer Antwort implementiert.
- „ResponseAdaptiveUrn“: hier wird die Urne verwaltet. Im Gegensatz zu dem Urnenmodell nach Wei, kann diese zu Beginn leer sein.
- „ResponseAdaptiveTempData“: hier werden die temporären Daten (die Urnen) verwaltet

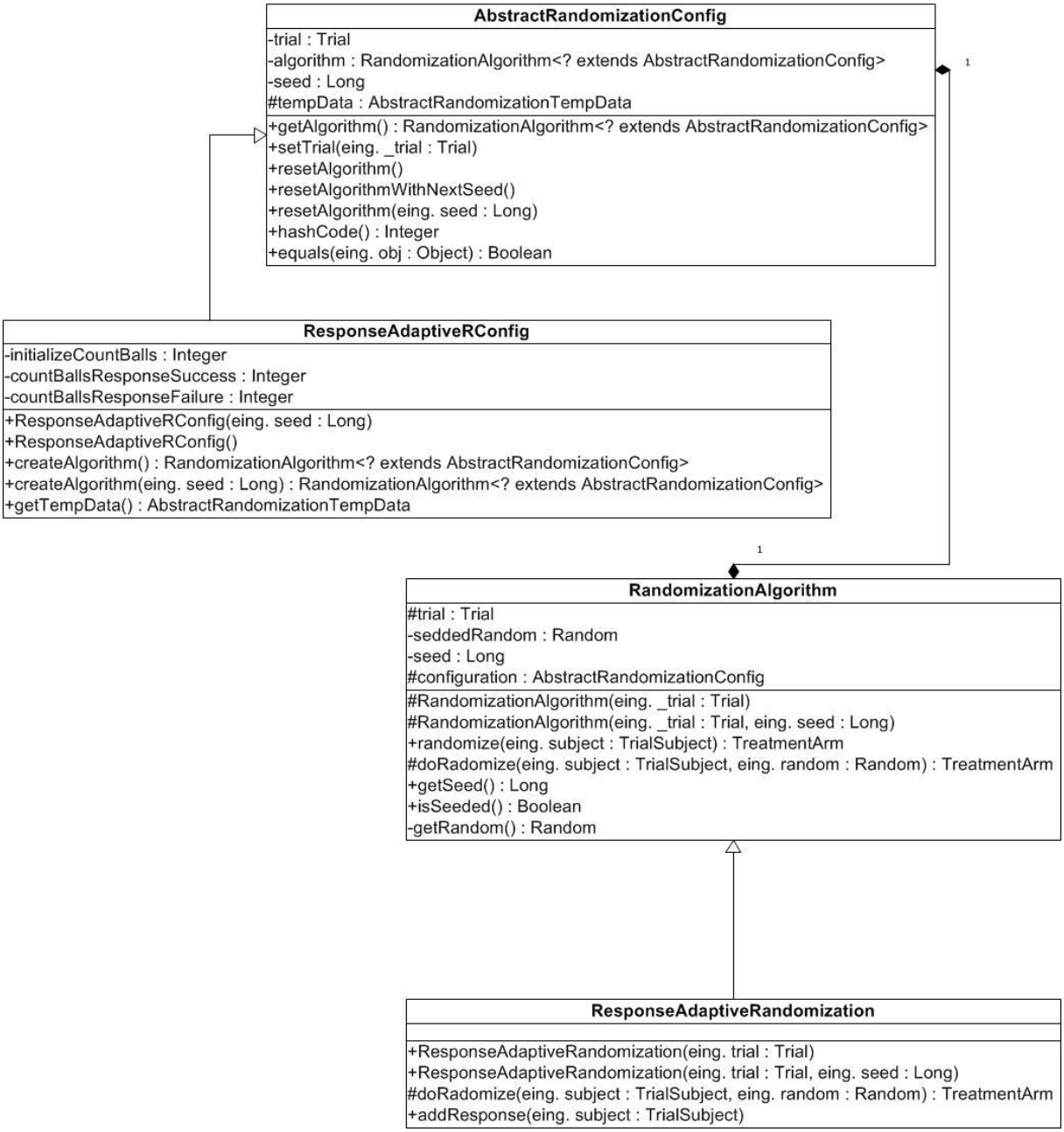

Abbildung 4.2. 1: Klassendiagramm: Response-adaptive Randomisation (Teil 1)

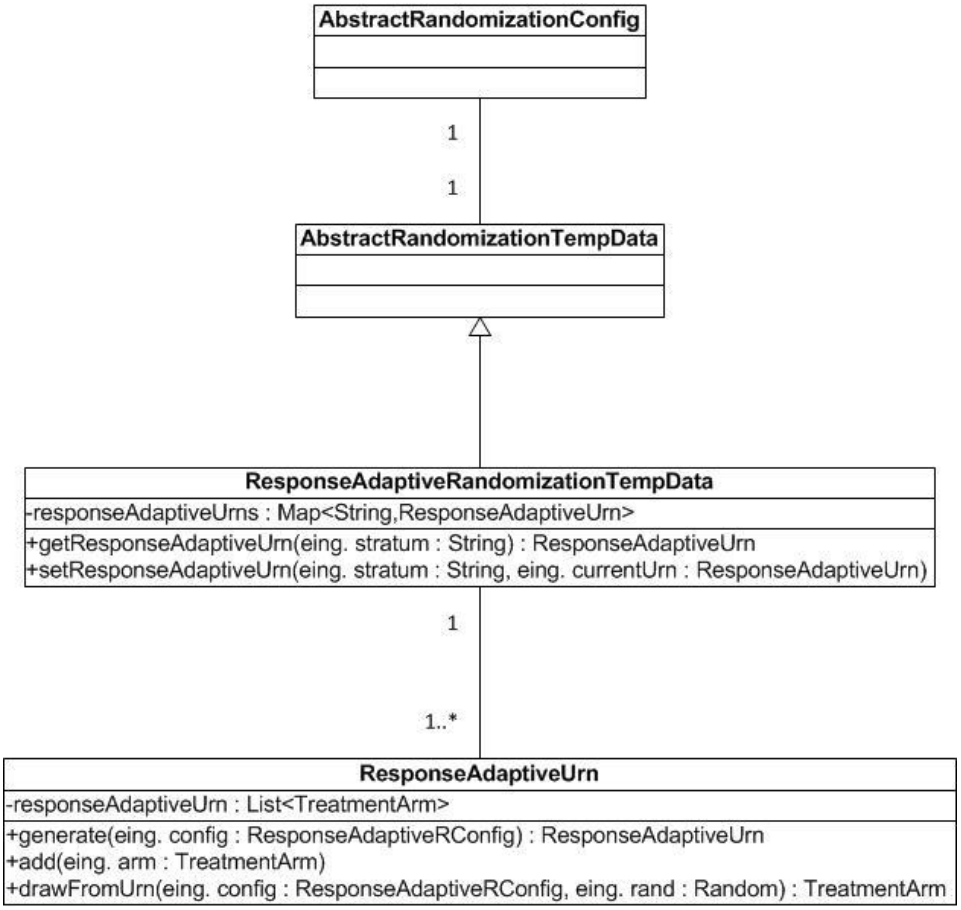

Abbildung 4.2. 2: Klassendiagramm: Response-adaptive Randomisation (Teil 2)

Die Abbildungen 4.2.1 und 4.2.2 zeigen die Zusammenhänge und Beziehungen zwischen den neuen *(ResponseAdaptiveRConfig, ResponseAdaptiveRandomization, ResponseAdaptiveRandomizationTempData und ResponseAdaptiveUrn)* und den alten *(AbstractRandomizationConfig, RandomizationAlgorithm und AbstractRandomizationTempData)* Klassen und deren Aufbau. Die Klasse *ResponseAdaptiveRConfig* erbt von der Klasse *AbstractRandomizationConfig* und implementiert die abstrakten Methoden der übergeordneten Klasse. Im Vergleich zu anderen implementierten Randomisationsverfahren wird die Klasse *ResponseAdaptiveRandomization* um eine neue Methode „addResponse“ für die Aufnahme des Ergebnisses einer Behandlung erweitert. Dort wird die Aktualisierung der Urne (Objekt der Klasse *ResponseAdaptiveUrn*) nach Eingabe einer neuen Antwort verwaltet. Die Urne kann zum Beginn der Randomisation leer sein. In diesem Fall wird die Zuordnung zu einer Behandlung nicht durch Ziehen aus der Urne, sondern durch Ziehen einer zufälligen Zahl, bestimmt.

Das in Abbildung 4.2.3 dargestellte Aktivitätsdiagramm zeigt das Speichern einer neuen Antwort. Die Antwort kann nur gespeichert werden, wenn das übergegebene *TrialSubject*-Objekt ungleich null und in der Datenbank gespeichert ist. Außerdem muss das Attribut *responseProperty* ungleich null sein. Im Schritt 1.1 wird überprüft, ob die Studie, zu der dieses *TrialSubject* gehört, ein Antwort-adaptives Verfahren als Randomisationsalgorithmus nutzt. Nur unter dieser Voraussetzung wird die Methode weiter ausgeführt. Wie die Methode 1.3: addResponse(TrialSubject subject) aufgebaut ist, ist im Kapitel 5.1 detailliert beschrieben.

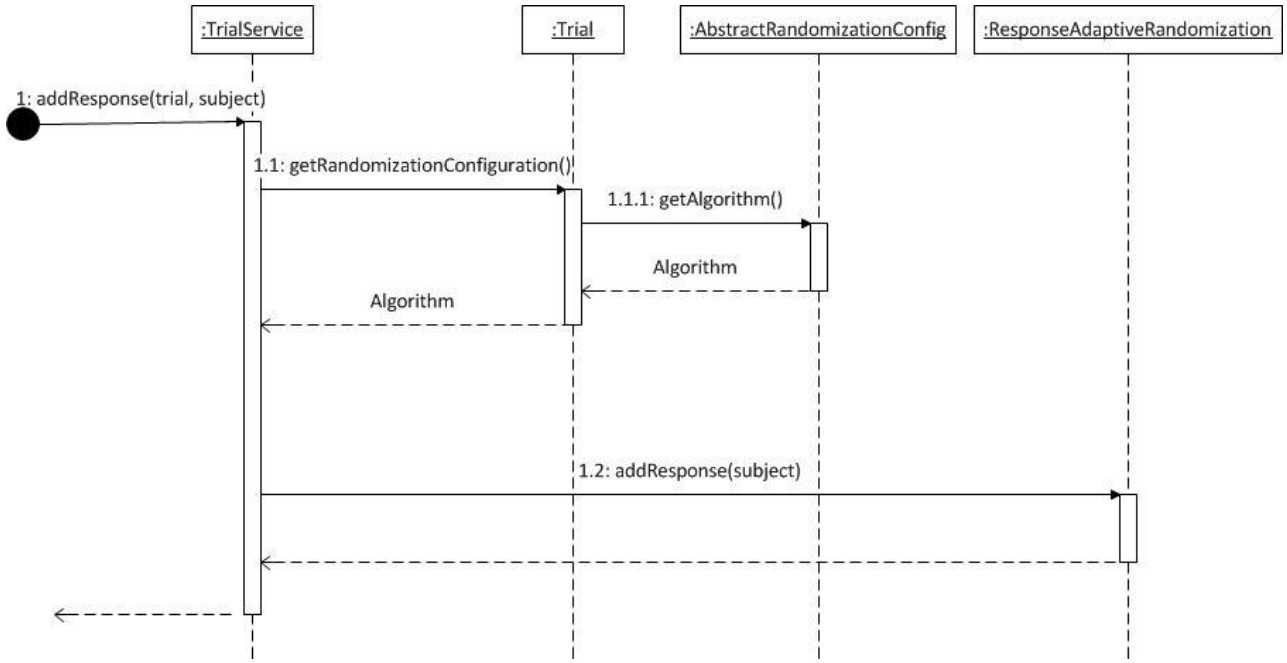

Abbildung 4.2. 3: Aktivitätsdiagramm: Hinzufügen einer Antwort

5. Implementierung

5.1. Implementierung des Algorithmus

Als Vorbereitung zur Implementierung des Algorithmus wurden sowohl die Klasse *"TrialSubject"* (Abbildung 5.1.1) als auch die Klasse *"Trial"* (Abbildung 5.1.2) um die entsprechenden Attribute erweitert. Diese Erweiterungen stellen die Voraussetzung für die Speicherung der Rückantwort dar.

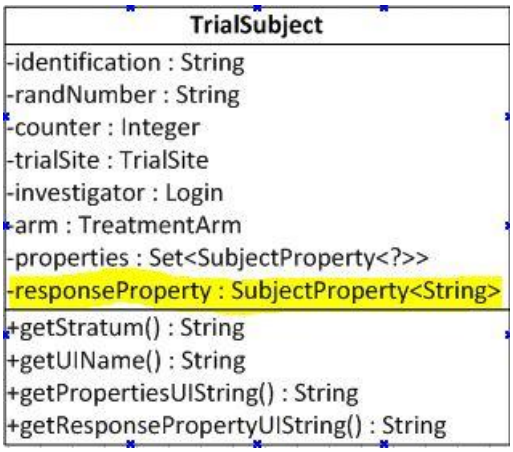

Abbildung 5.1. 1: Erweiterung der Klasse "TrialSubject"

Abbildung 5.1. 2: Erweiterung der Klasse "Trial"

Der Algorithmus wurde, wie auf den Abbildungen 4.2.1 und 4.2.2 dargestellt ist, implementiert.

Die Konfigurationsklasse *„ResponseAdaptiveRConfig“* beinhaltet drei ganzzahlige Variablen (int):

- *initializeCountBallsResponseAdaptiveR:* Anfangsballanzahl jeder Therapiemöglichkeiten in der Urne
- *countBallsResponseSuccess:* Anzahl der Bälle vom Typ t, die nach der erfolgreich abgeschlossenen Therapie t in die Urne hineingelegt werden sollen.
- *countBallsResponseFailure:* Anzahl der Bälle von allen anderen Typen außer dem Typ t, die nach der nicht erfolgreich abgeschlossenen Therapie t in die Urne hineingelegt werden sollen.

Wie im Kapitel 2.3.4 beschrieben ist, stehen diese Attribute in Abhängigkeit zueinander und zu der Anzahl der Behandlungsarme. Diese Abhängigkeiten werden in der Klasse *ResponseAdaptiveRandomizationConfigValidator* auf Gültigkeit überprüft (Abbildung 5.1.3). Durch die Annotation

```
@Constraint (validatedBy=ResponseAdaptiveRandomizationConfigValidator.class)
```

wird die Verbindung zwischen dem Validator und der Schnittstelle *ResponseAdaptiveRandomizationConfigA* hergestellt. Der Bean *ResponseAdaptiveRConfig* enthält bei seiner Definition die Annotation dieser Schnittstelle, wodurch die richtige Validierung der Daten gewährleistet wird.

```
@Override
public boolean isValid(ResponseAdaptiveRConfig config,
        ConstraintValidatorContext constraintContext) {

    int countBallsResponseFailure = config.getCountBallsResponseFailure();
    int countBallsResponseSuccess = config.getCountBallsResponseSuccess();
    int countTreatmentArms = config.getTrial().getTreatmentArms().size();

    return ((countBallsResponseFailure * (countTreatmentArms - 1) >= 0)
            && (countBallsResponseFailure * (countTreatmentArms - 1) <= countBallsResponseSuccess)
            && (countBallsResponseFailure % (countTreatmentArms - 1) == 0) && (countBallsResponseSuccess
            % (countTreatmentArms - 1) == 0));
}
```

Abbildung 5.1. 3: **Codeabschnitt: Methode der Klasse *ResponseAdaptiveRandomizationConfigValidator*, die die Abhängigkeiten zwischen den Attributen und der Anzahl der Behandlungsarme überprüft**.

In der Klasse *ResponseAdaptiveRandomization* sind zwei wichtigen Funktionen implementiert: Die Randomisierung eines neuen Patienten und das Hinzufügen einer neuen Antwort. Die Randomisierung findet, genauso wie bei allen anderen Algorithmen, in der Methode *doRandomize* statt (Abbildung 5.1.4). Bei der Randomisation wird aus der Urne eine der Behandlungsmöglichkeiten mit Zurücklegen gezogen und zurückgegeben.

```
@Override
protected TreatmentArm doRadomize(TrialSubject subject, Random random) {
    ResponseAdaptiveRandomizationTempData tempData =
        (ResponseAdaptiveRandomizationTempData) super.configuration.getTempData();
    String stratum = "";
    if(trial.isStratifyTrialSite())
        stratum = subject.getTrialSite().getId() + "__";
    stratum += subject.getStratum();
    ResponseAdaptiveUrn responseAdaptiveUrn = tempData.getResponseAdaptiveUrn(stratum);
    if (responseAdaptiveUrn == null) {
        responseAdaptiveUrn = ResponseAdaptiveUrn.generate(configuration);
        tempData.setResponseAdaptiveUrn(stratum, responseAdaptiveUrn);
    }
    TreatmentArm drawnArm = responseAdaptiveUrn.drawFromUrn(configuration,random);
    return drawnArm;
}
```

Abbildung 5.1. 4: **Codeabschnitt: Realisierung der Methode doRandomize()**

Das Hinzufügen einer Rückantwort kann in Abhängigkeit, ob das Ergebnis positiv oder negativ ist, auf zwei unterschiedlichen Weisen erfolgen (Abbildung 5.1.5).

1. Im Fall einer positiven Rückantwort (*option1*) werden Bälle des Typs der Behandlungsart, die eingesetzt wurde, in die Urne dazugelegt. Anzahl der Bälle ist genauso groß, wie bei der Konfiguration des Algorithmus im Feld „Anzahl der zurückgelegten Bälle bei **erfolgreich** abgeschlossener Behandlung“ eingegeben wurde. Gleichzeitig werden Bälle anderen Behandlungsarten auch dazugelegt. Anzahl der Bälle von jeder Art ist genauso groß , wie bei der Konfiguration des Algorithmus im Feld „Anzahl der zurückgelegten Bälle bei **nicht erfolgreich** abgeschlossener Behandlung“ eingegeben wurde, dividiert durch Anzahl der Behandlungsarme minus eins.
2. Im Fall einer negativen Rückantwort (*option2*) werden Bälle des Typs der Behandlungsart, die eingesetzt wurde, in die Urne dazugelegt. Anzahl der Bälle ist genauso groß, wie bei der Konfiguration des Algorithmus im Feld „Anzahl der zurückgelegten Bälle bei **nicht erfolgreich** abgeschlossener

Behandlung“ eingegeben wurde. Gleichzeitig werden Bälle anderen Behandlungsarten auch dazugelegt. Anzahl der Bälle von jeder Art ist genauso groß , wie bei der Konfiguration des Algorithmus im Feld „Anzahl der zurückgelegten Bälle bei **erfolgreich** abgeschlossener Behandlung“ eingegeben wurde, dividiert durch Anzahl der Behandlungsarme minus eins.

Die Variable *arm* stellt die Therapie dar, nach welcher Durchführung eine Rückantwort gespeichert werden soll. Die Liste *arms* beinhaltet zuerst alle zu der Studie dazugehörigen Therapien. Die Therapie *arm* wird aus dieser Liste nachträglich gelöscht.

```
List<TreatmentArm> arms = new ArrayList<TreatmentArm>(trial.getTreatmentArms());
arms.remove(arm);
String response = subject.getResponseProperty().getValue();
if (response.equals(trial.getTreatmentResponse().getOption1())) {
    for (int i = 0; i < configuration.getCountBallsResponseSuccess(); i++) {
        responseAdaptiveUrn.add(arm);
    }
    int countBallsOtherArms = configuration
            .getCountBallsResponseFailure() / (arms.size());
    for (TreatmentArm tArm : arms) {
        for (int i = 0; i < countBallsOtherArms; i++) {
            responseAdaptiveUrn.add(tArm);
        }
    }
} else {
    for (int i = 0; i < configuration.getCountBallsResponseFailure(); i++) {
        responseAdaptiveUrn.add(arm);
    }
    int countBallsOtherArms = configuration
            .getCountBallsResponseSuccess() / (arms.size());
    for (TreatmentArm tArm : arms) {
        for (int i = 0; i < countBallsOtherArms; i++) {
            responseAdaptiveUrn.add(tArm);
        }
    }
}
```

Abbildung 5.1. 5: Codeabschnitt: Hinzufügen einer Rückantwort

Die Verwaltung einer Urne ist in der Klasse *ResponseAdaptiveUrn* realisiert. Diese Klasse bietet drei Funktionalitäten:

- Erzeugen einer Urne: Hier wird eine Urne, sollte noch keine existieren, erzeugt. Wenn die Anfangsballanzahl (*initializeCountBallsResponseAdaptiveR)* größer als 0 ist, dann werden so viele „Bälle“ von jeder der Therapiemöglichkeiten in diese Urne hineingelegt

(Kapitel 2.3.4-> Stratifizierte und randomiesierte play-the-Winner (SRPW) Regel)

- Hinzufügen eines „Balles“ , der eine Therapie repräsentiert
- Zufällige Ziehung einer Therapiemöglichkeit aus der Urne mit Zurücklegen. Falls die Anfangsballanzahl gleich 0 ist, ist die Urne zu dem Moment der ersten Ziehung leer. In diesem Fall wird eine Zufällige Zahl zwischen 0 und 1 generiert (Abbildung 5.1.6) und in Abhängigkeit, wie groß die Zahl ist, wird eine der Therapiemöglichkeiten ausgewählt (Kapitel 2.3.4).

```
TreatmentArm drawnArm = new TreatmentArm();
double randomNumber = rand.nextDouble();
Set<TreatmentArm> arms = config.getTrial().getTreatmentArms();
int  i=0;
for(TreatmentArm arm: arms){
    i++;
    if((randomNumber<(i/arms.size())) && (randomNumber>=(i-1)/arms.size())){
        drawnArm=arm;
        break;
    }
}
return drawnArm;
```

Abbildung 5.1. 6: **Codeabschnitt: Realisierung des zufälligen Ziehens aus der Urne, wenn sie leer ist.**

5.2. Implementierung der Oberfläche für die Algorithmuskonfiguration

Nachdem der Algorithmus fertig implementiert worden ist, wurde die Oberfläche für seine Konfiguration realisiert. Bei der Konfiguration werden folgende Eingaben vom Benutzer erwartet (Abbildung 5.2.1):

1. Anfangsballanzahl (pro Behandlung): Für jede Behandlungsmöglichkeit werden zum Beginn so viele Bälle, wie der Benutzer in diesem Feld eingibt, in die Urne hineingelegt.
2. Anzahl der zurückgelegten Bälle bei erfolgreich abgeschlossener Behandlung: Nach der Eingabe des positiven Ergebnisses für eine schon durchgeführte Behandlung werden so viele Bälle, die diese Behandlung darstellen, in die Urne hineingelegt. Gleichzeitig werden Bälle, die alle anderen Behandlungen darstellen, auch in die Urne hineingelegt. Die Anzahl der Bälle, die jeweils eine Behandlung abbilden, wird folgendermaßen berechnet: Inhalt aus dem Feld 3 „Anzahl der

zurückgelegten Bälle bei nicht erfolgreich abgeschlossener Behandlung“ / (Anzahl der im Schritt 3 eingegebenen Behandlungsarme minus eins).

3. Anzahl der zurückgelegten Bälle bei nicht erfolgreich abgeschlossener Behandlung: Nach der Eingabe des negativen Ergebnisses für eine schon durchgeführte Behandlung werden so viele Bälle, die diese Behandlung darstellen, in die Urne hineingelegt. Gleichzeitig werden Bälle, die alle anderen Behandlungen darstellen, auch in die Urne hineingelegt. Die Anzahl der Bälle, die jeweils eine Behandlung abbilden, wird folgender Maßen berechnet: Inhalt aus dem Feld 2 „Anzahl der zurückgelegten Bälle bei erfolgreich abgeschlossener Behandlung“ /(Anzahl der im Schritt 3 eingegebenen Behandlungsarme minus eins).

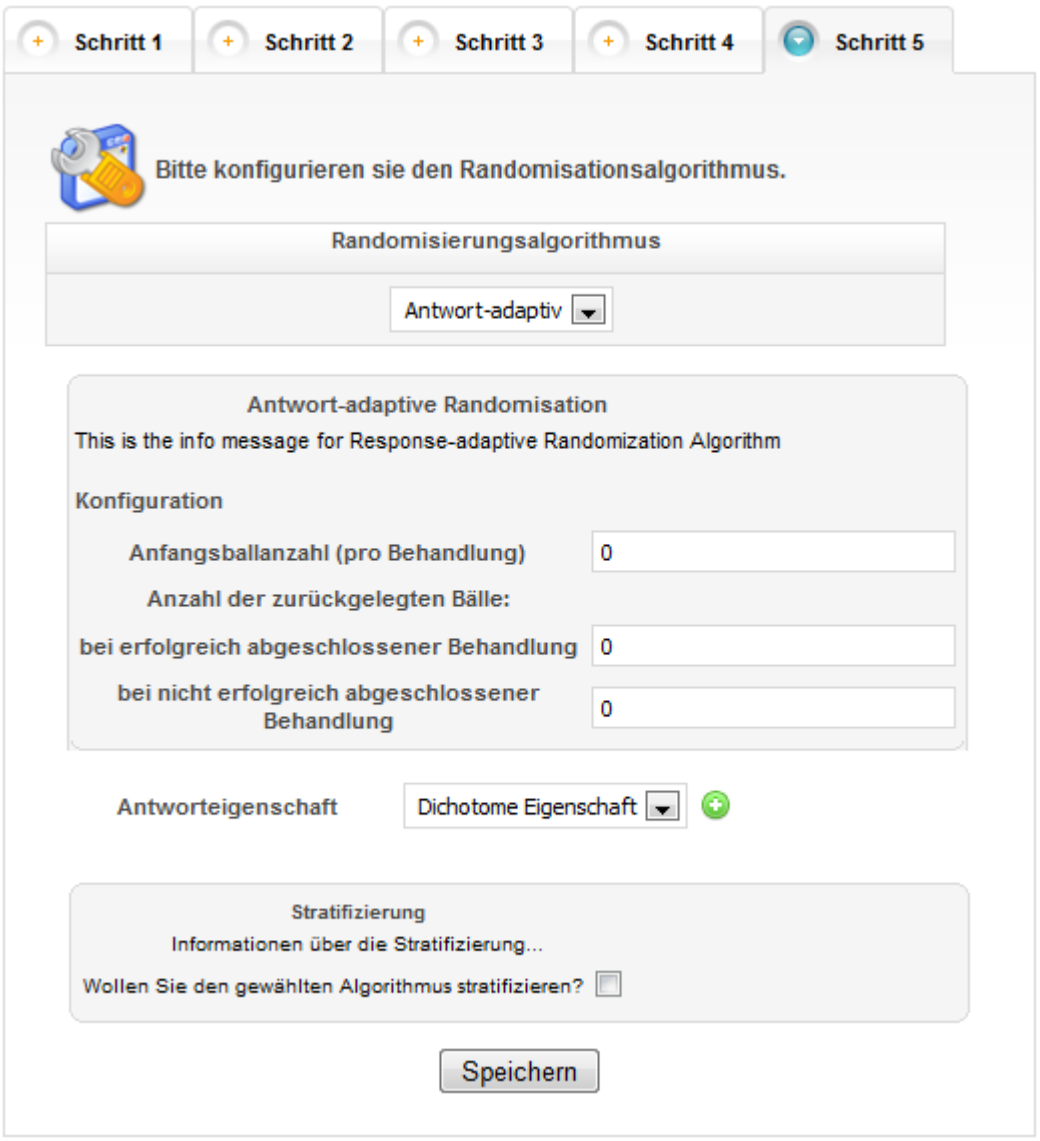

Abbildung 5.2. 1: Oberfläche für Konfiguration des antwort-adaptiven Randomisationsalgorithmus.

Die Antwort-Eigenschaft kann in der Form der dichotomen Eigenschaft eingegeben werden. Nachdem der Benutzer das grüne Plussymbol anklickt, erscheint die Eingabemaske(Abbildung 5.2.2) mit der Möglichkeit den Namen, die Beschreibung und die zwei Möglichkeiten des Ergebnisses der Antwort einzugeben. Das Plussymbol wird nach dem Anklicken unsichtbar, da bei einer Studie nur eine Antworteigenschaft eingegeben werden kann.

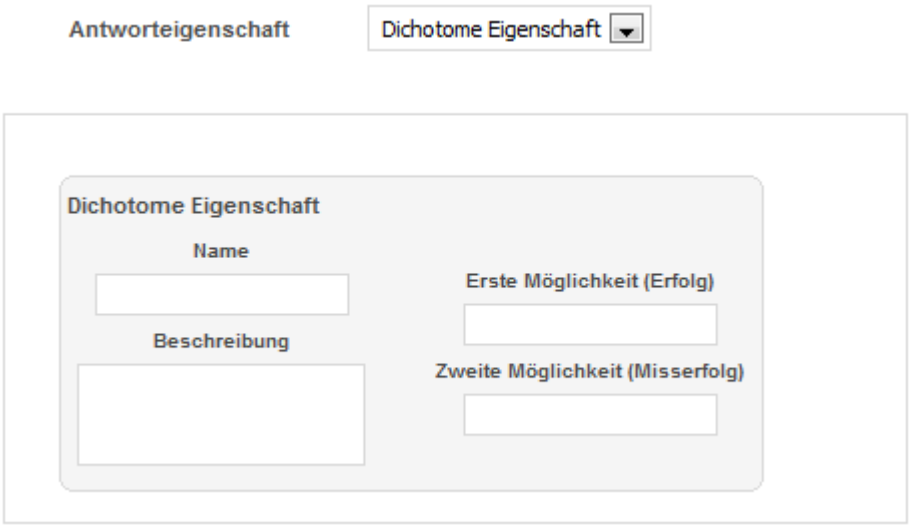

Abbildung 5.2. 2: Eingabemaske für die Antwort-Eigenschaft

Nachdem eine Studie erfolgreich angelegt wurde, kann der Benutzer alle eingetragenen Konfigurationen in einem Übersichtsfenster ansehen (Abbildung 5.2.3)

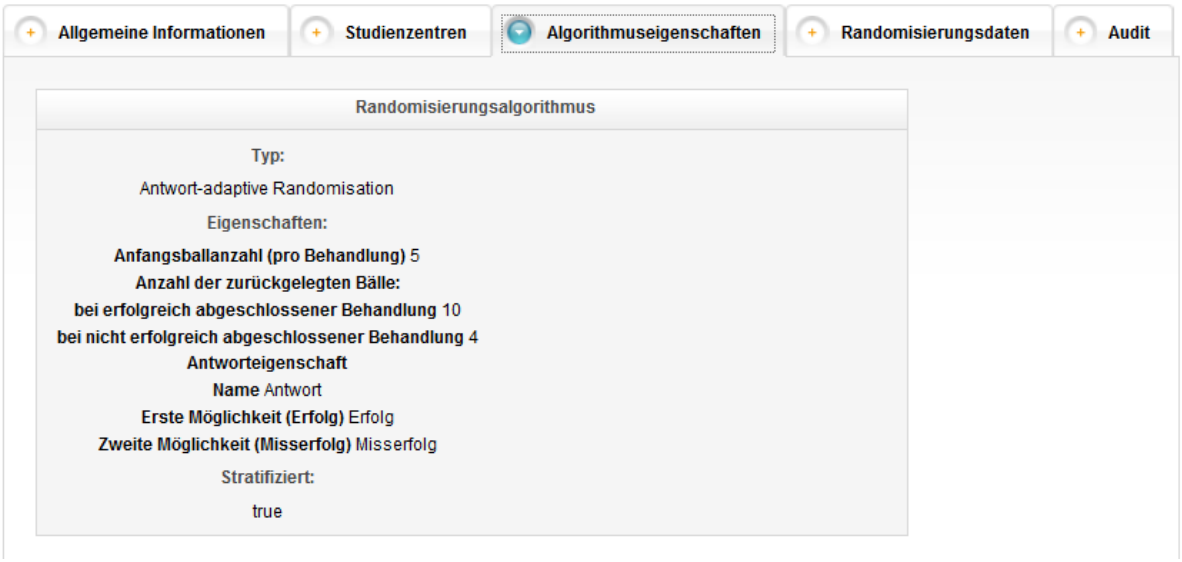

Abbildung 5.2. 3: Übersicht der Konfigurationen, die bei der Erstellung der Studie eingegeben wurden

5.3. Implementierung der Oberfläche für die Antworteingabe

Als Erstes wurde für die Antworteingabe das Menü um einen neuen Menüpunkt „Antwort hinzufügen“ erweitert. Dieser Menüpunkt wird nur angezeigt, wenn der Benutzer Rolle „Prüfarzt“ hat und eine aktive Studie ausgewählt hat, bei der er die benötigte Rechte besitzt. Diese Studie muss als Randomisationsverfahren „Antwort-adaptive Randomisation“ haben. Nach der Auswahl des neuen Menüpunktes erscheint auf dem Bildschirm eine Oberfläche für die Identifizierung des Probanden (Abbildung 5.3.1). Voraussetzung für eine erfolgreiche Identifizierung ist das Vorhandensein eines gültigen IDs. Da für jeden Probanden nur eine Antwort gespeichert werden darf, erscheint eine Fehlermeldung, falls der Benutzer die ID eines Probandes mit schon eingetragener Antwort eingibt (Abbildung 5.3.2).

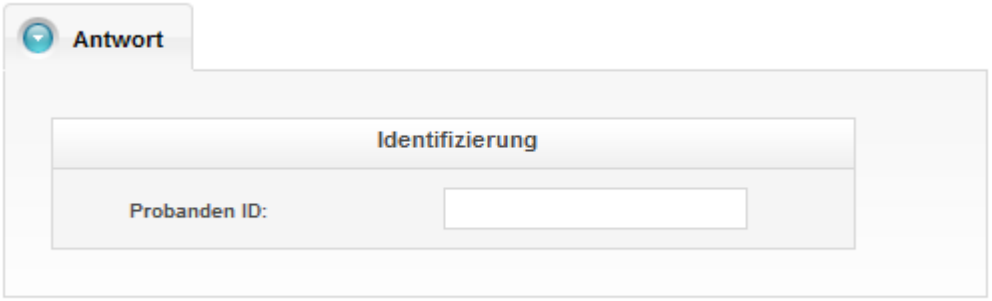

Abbildung 5.3. 1: Oberfläche für die Identifizierung des Probanden für die Antworteingabe

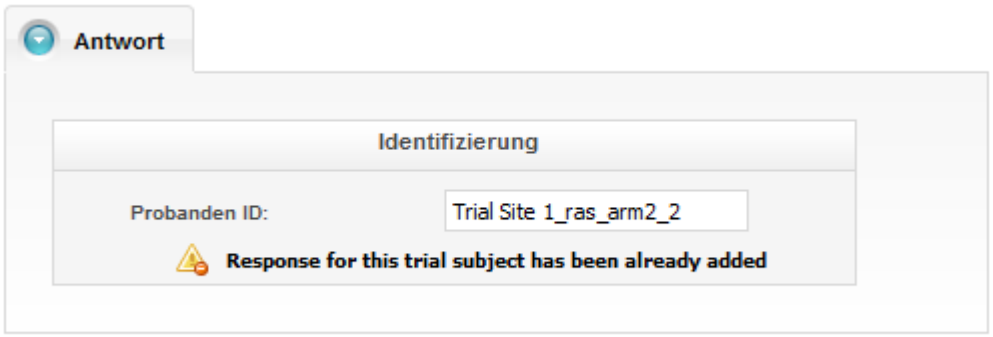

Abbildung 5.3. 2: Fehlermeldung bei dem Versuch wiederholter Eingabe der Antwort.

Nachdem der Benutzer die Probanden ID erfolgreich eingegeben hat, wird eine kurze Information über den Probanden angezeigt, um sicherzugehen, ob der richtige Patient ausgewählt wurde. Diese Information beinhaltet die Studie, an der der Patient teilnimmt, den Behandlungsarm und die Eigenschaften des Patienten. Danach hat der Benutzer die Möglichkeit die Antworteigenschaft einzutragen und zu speichern (Abbildung 5.3.3). Anschließend bekommt er eine Bestätigung über das erfolgreiche Speichern durch eine Popup-Meldung (Abbildung 5.3.4) angezeigt.

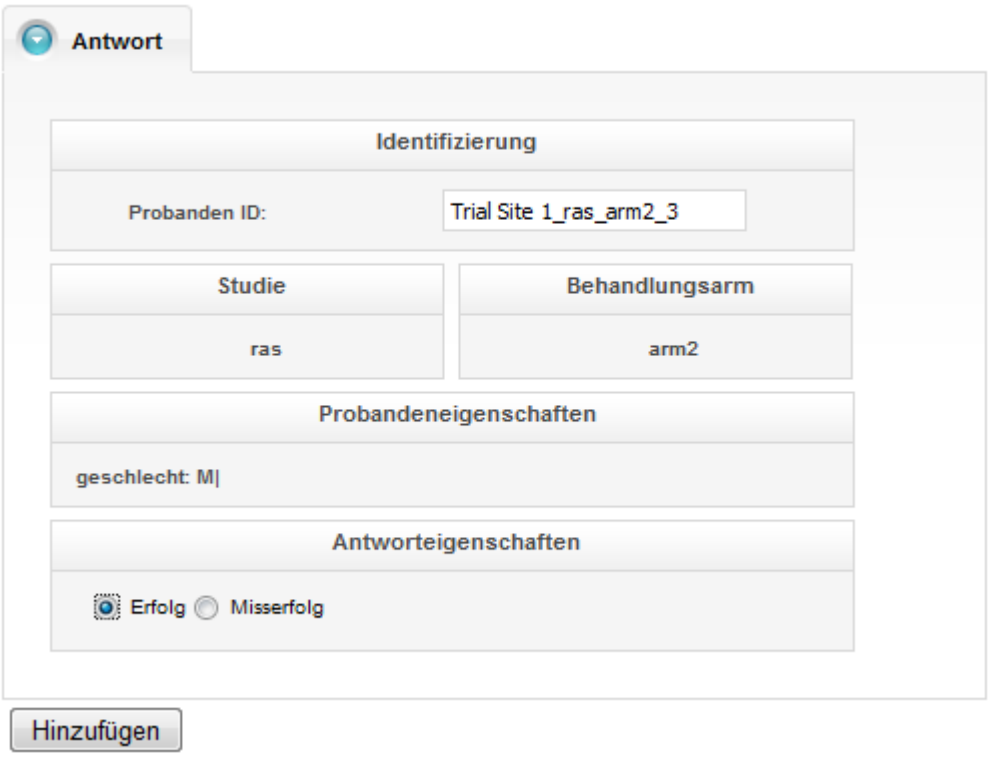

Abbildung 5.3. 3: Oberfläche zur Eingabe der Antwort

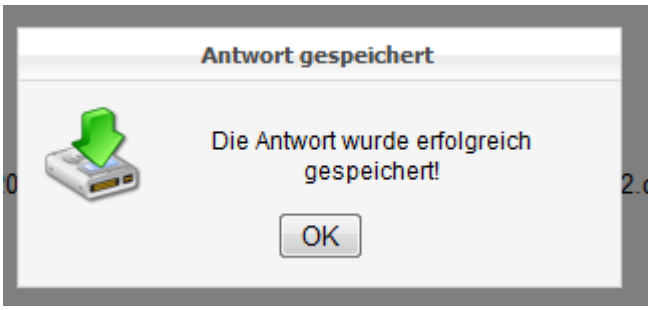

Abbildung 5.3. 4: Popup Meldung nach der erfolgreichen Eingabe einer Antwort

6. Test

Die Implementierung wird auf zwei unterschiedliche Arten getestet:

- Die Implementierung des Algorithmus wird mit Hilfe von JUnit-Tests überprüft.
- Die Oberfläche wird durch die Eingabe der korrekten bzw. falschen Daten und Dokumentation der Ergebnisse überprüft.

Die JUnit-Tests dienen zum Testen der Ausführung des Randomisationsalgorithmus. Sie befinden sich in dem Paket „src/test/java/de.randi2.core.unit“. Dort ist für jede existierende Java-Klasse eine JUnit-Testklasse zu finden. Diese JUnit-Klasse implementiert die Tests für jede Funktionalität der zu testenden Klasse.

Die Oberflächentests werden wie in den folgenden Unterkapiteln definiert und durchgeführt.

6.1. Anlegen einer response-adaptiven Studie

In diesem Testfall soll das korrekte Anlegen einer response-adaptiven Studie überprüft werden.

Vorbereitung: Für die Testdurchführung soll die Entwicklungsumgebung vorbereitet werden. Dies ist ausführlich auf der Randi2 Wiki-Seite beschrieben[13]. Während dieser Vorbereitung wird die Datenbank mit allen notwendigen Testdaten befüllt (Schritt 12 – Ausführen der Klasse *Bootstrap*). Für diesen Testfall sind wichtig:

- Benutzer mit der Rolle „Studienleiter"
  (Benutzername: p_investigator, Passwort: 1$heidelberg)
- teilnehmende Studienzentren
- Studienleiter
- Benutzer mit der Rolle „Prüfarzt“
  (Benutzername: investigator, Passwort: 1$heidelberg)

[13] http://www.randi2.org/retro/wiki/DeveloperSetup

Testfall 1: Erfolgreiches Anlegen einer response-adaptiven Studie mit zwei Behandlungsarmen.

Zum erfolgreichen Anlegen einer Studie müssen die fünf Schritte, die im Kapitel 3.1 beschrieben sind, gemacht werden. Im Schritt 3 sollen zwei Behandlungsarme angelegt werden. Im Schritt 4 soll eine dichotome Eigenschaft „Geschlecht“ mit den Werten „m“ und „w“ angelegt werden. Um eine Antwort-adaptive Studie anlegen zu können, soll im Schritt fünf als Randomisierungsalgorithmus „Antwort-adaptiv“ gewählt werden. Für das erfolgreiche Anlegen einer response-adaptiven Studie sollen die zwei folgenden Konfigurationsbedingungen erfüllt werden:

1. Anzahl der zurückgelegten Bälle bei erfolgreich abgeschlossener Behandlung ist vielfaches von der Anzahl der Behandlungsarme -1.
2. Anzahl der zurückgelegten Bälle bei nicht erfolgreich abgeschlossener Behandlung ist vielfaches von der Anzahl der Behandlungsarme -1.

Bei einer Studie mit zwei Behandlungsarmen sind diese zwei Bedingungen immer erfüllt und die Studie kann somit erfolgreich angelegt werden. Die Abbildung 6.1.1 zeigt den Screenshot mit den Eingaben für diesen Test.

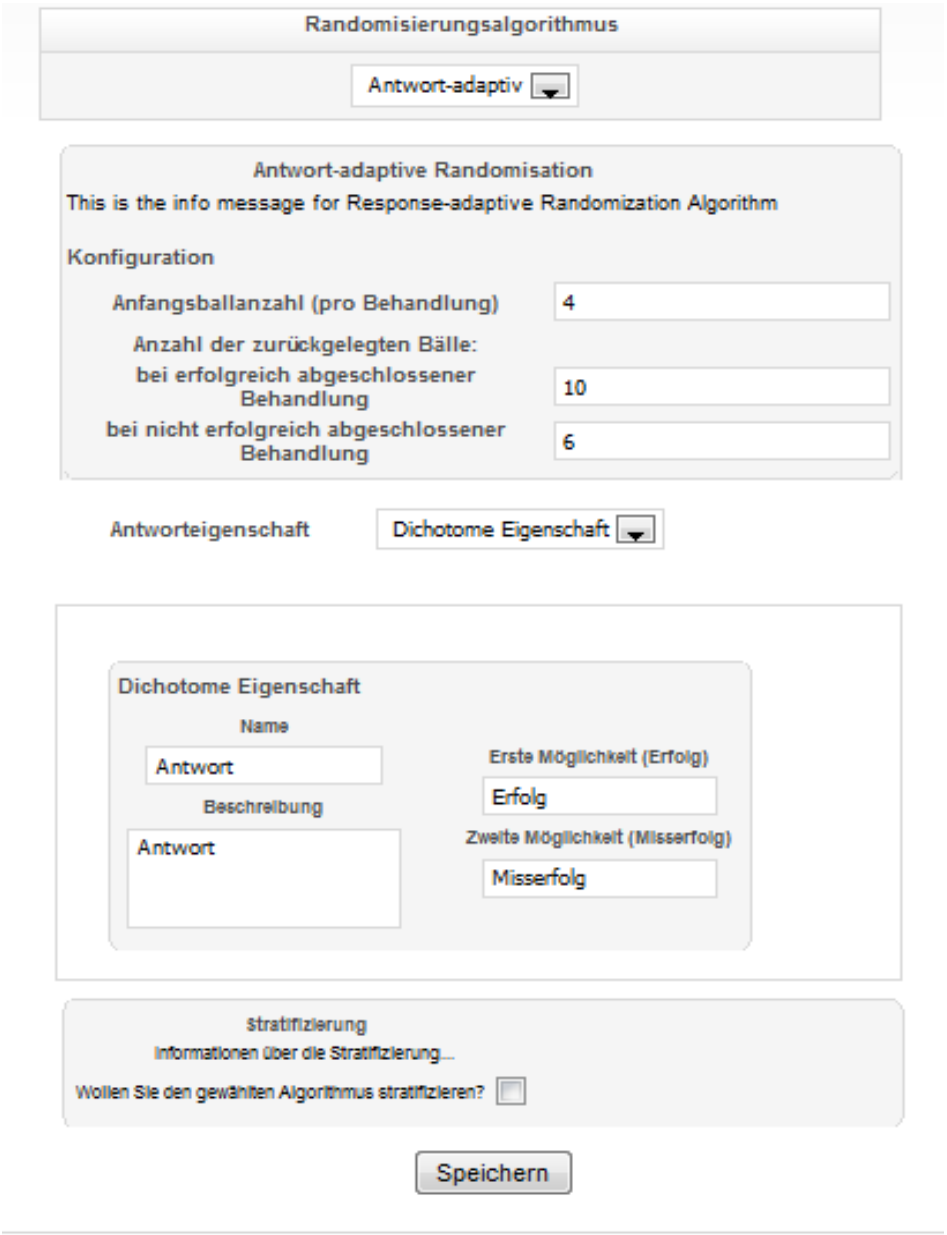

Abbildung 6.1. 1: Durchführung des Testfalls 1.

Als Bestätigung der Aufnahme der Studie in die Datenbank dient die Meldung über das erfolgreiche Anlegen.

Testfall 2: Erfolgreiches Anlegen einer response-adaptiven Studie mit drei Behandlungsarmen.

Beim Anlegen dieser Studie sollen im Schritt 3 drei Behandlungsarme angelegt werden. Weiterhin soll genauso wie im Testfall 1 im Schritt 4 beim Anlegen einer Studie eine Dichotome Eigenschaft „Geschlecht“ angelegt werden und im Schritt 5 soll „Antwort-adaptiv“ als Randomisationsalgorithmus gewählt werden. Im Gegensatz zum Testfall 1 soll bei diesem Test auf die Anzahl der zurückgelegten Bälle geachtet

werden, damit die Verhältnisse 1 und 2 aus dem Testfall 1 stimmen. Das heißt, sowohl die Anzahl der zurückgelegten Bälle bei einer erfolgreich als auch nicht erfolgreich abgeschlossenen Behandlung soll teilbar durch zwei sein (3 Behandlungsarme minus 1). Die Eingabe von den gleichen Konfigurationsparametern wie im Testfall 1 führt zu erfolgreicher Aufnahme dieser Studie in die Datenbank. Dieser Studie soll nach der Eigenschaft „Geschlecht“ stratifiziert werden.

Testfall 3: Anlegen einer response-adaptiven Studie mit drei Behandlungsarmen und nicht korrekten Konfigurationsdaten

Um diesen Test durchführen zu können soll analog dem Testfall 2 eine Antwort-adaptive Studie angelegt werden. Um die Korrekte Validierung der Konfigurationsdaten zu überprüfen, sollen falsche Daten eingetragen werden, mit der Erwartung eine Fehlermeldung zu erhalten. Um die Fehlermeldung zu bekommen, können folgende Daten eingegeben werden:

- Anzahl der zurückgelegten Bälle bei erfolgreich abgeschlossener Behandlung nicht durch 2 teilbar
- Anzahl der zurückgelegten Bälle bei nicht erfolgreich abgeschlossener Behandlung nicht durch 2 teilbar
- Anzahl der zurückgelegten Bälle bei erfolgreich abgeschlossener Behandlung kleiner als Anzahl der zurückgelegten Bälle bei nicht erfolgreich abgeschlossener Behandlung

Die Abbildung 6.1.2 zeigt den Screenshot mit den Eingaben für diesen Test.

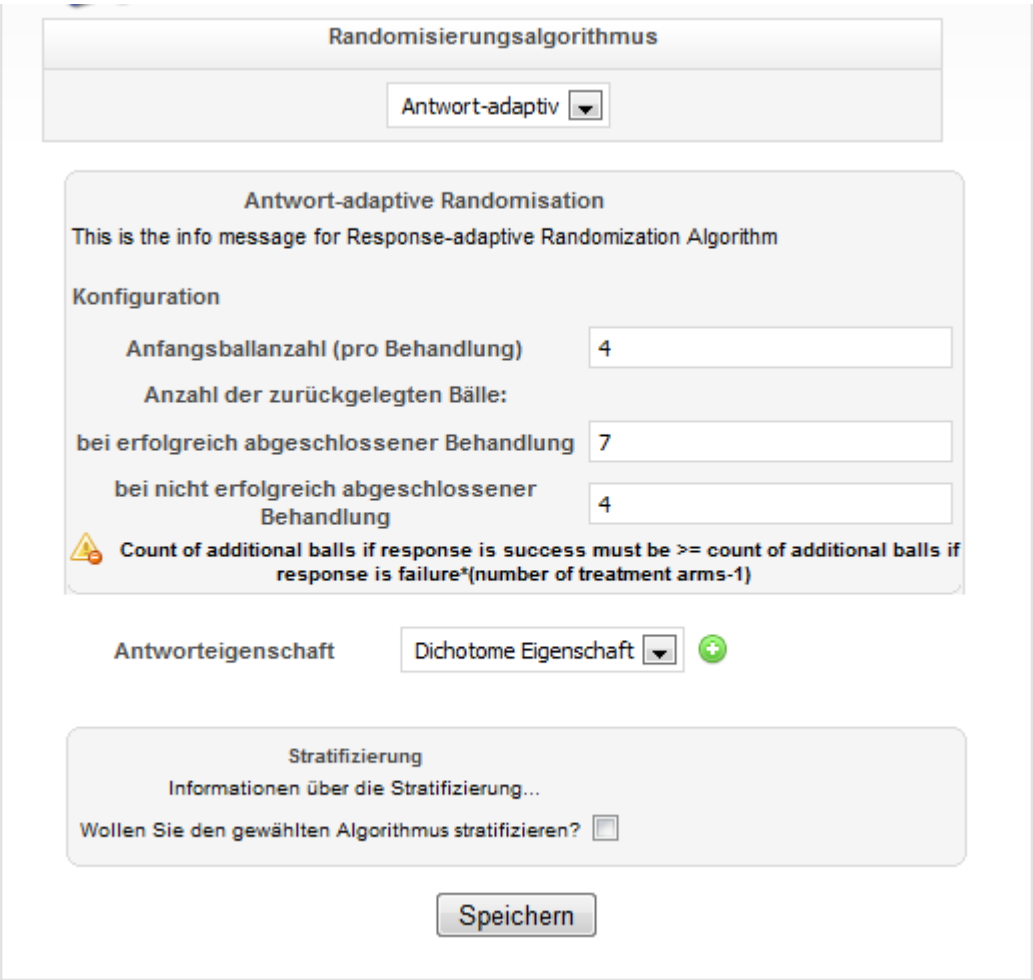

Abbildung 6.1. 2: Durchführung des Testfalls 3.

6.2. Randomisation eines neuen Patienten

In diesem Testfall soll das korrekte Randomisieren eines Patienten überprüft werden. Außer, dass der Patient am Ende des Tests erfolgreich zu einem Behandlungsarm aufgenommen werden soll, soll die betroffene Urne in der Datenbank richtig aktualisiert werden.

Vorbereitung: Als Vorbereitung für diesen Test soll der Testfall 2 aus Kapitel 6.1 durchgeführt werden. Anschließend soll der Benutzer in der Rolle „Studienleiter“ den Status der in diesem Testfall angelegter Studie auf „Aktiv“ setzen. Dafür meldet er sich an und wählt im Menüpunkt „Studie“ Untermenüpunkt „Meine Studien“. Danach soll er bei der gerade erstellten Studie den Link „Anzeigen“ klicken und dann den Button „Bearbeiten“ drücken, den Status im Dropdown Menü auf „Aktiv“ wechseln und den Button „Speichern“ drücken. Anschließend kann sich der Benutzer in der Rolle „Prüfarzt“ anmelden und den Patienten in dieser Studie randomisieren.

Durchführung: Der Benutzer mit der Rolle „Prüfarzt“ meldet sich an und wählt den Menüpunkt „Meine Studien“. Dann wählt er von der Liste seiner gesamten Studien, die im Testfall 2 des Kapitels 6.1 erstellte Studie, aus. Nachdem er die Studie ausgewählt hat, erscheint im Menü ein neuer Menüpunkt „Proband“ mit zwei Untermenüpunkten:

- Proband hinzufügen
- Antwort hinzufügen

Nach dem Auswahl des Menüpunktes „Proband hinzufügen“ erscheint die Eingabemaske für die Randomisation eines neuen Patienten. In dieser Eingabemaske soll nur die dichotome Eigenschaft „Geschlecht“ des Patienten eingetragen werden. Nach dem Drücken auf den Button „Hinzufügen“ wird der Patient zu einer der Behandlungsarme zugewiesen. Die Patienten-ID und der zugewiesene Behandlungsarm erscheinen in der Bestätigungsmeldung. (Abbildung 6.2.1)

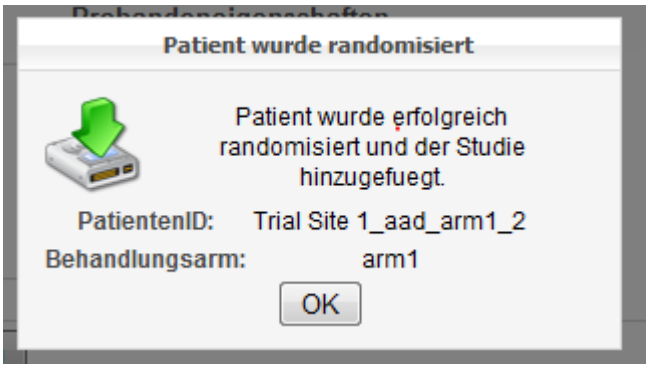

Abbildung 6.2. 1: Bestätigung der Aufnahme des Patienten zu einer der Behandlungsarme.

Da das Ziehen aus der Urne bei dem implementierten Algorithmus mit Zurücklegen funktioniert, soll die Anzahl der Bälle in der Urne nach der Randomisation unverändert bleiben. Laut der Konfiguration bedeutet das, dass in der Urne jeweils 4 Bälle jeder Therapieart vorhanden sein sollen. Die Anzahl der Bälle kann direkt aus der Datenbanktabelle „responseadaptiveurn_treatmentarm“ abgelesen werden, indem die Query:

SELECT * FROM responseadaptiveurn_treatmentarm where responseadaptiveurn_id = *responseadaptiveurn_id*

ausgeführt wird.

6.3. Eingabe einer neuen Antwort

In diesem Testfall soll die korrekte Eingabe einer Antwort überprüft werden. Außer, dass die Antwort am Ende des Tests erfolgreich in der Datenbank gespeichert werden soll, soll die betroffene Urne aktualisiert werden.

Vorbereitung: Als Vorbereitung für die Durchführung dieses Tests soll der Test aus dem Kapitel 6.2 durchgeführt werden. Anschließend soll sich der Benutzer mit der Rolle „Prüfarzt“ angemeldet bleiben und den Menüpunkt „Antwort hinzufügen“ auswählen. Nach der Auswahl erscheint im Hauptfenster die Eingabemaske für die Aufnahme einer neuen Antwort.

Testfall 1: Erfolgreiche Aufnahme einer neuen Antwort

Damit eine Antwort erfolgreich aufgenommen werden kann, soll in dem Feld „Probanden ID“ eine gültige ID eingegeben werden. Genommen wird die ID des im Testfall aus Kapitel 6.2 aufgenommenen Patienten (hier: Trial Site 1_aad_arm1_2). Nachdem der Patient erfolgreich identifiziert wurde, erscheinen seine Daten mit der Möglichkeit ein Therapieergebnis einzugeben. Nach der Auswahl der Checkbox für das erfolgreiche Beenden der Behandlung soll der Button „Hinzufügen“ gedrückt werden. Es erscheint eine Meldung über die erfolgreiche Aufnahme der Antwort.

Nach der erfolgreichen Aufnahme der Antwort soll die Urne aktualisiert werden. Da die Behandlung positiv beendet wurde, sollen 10 zusätzliche Bälle vom Typ dieser Behandlung in die Urne hineingelegt werden (zusammen sind es 14 - Anfangsballanzahl: 4 und zusätzlich: 10) und jeweils 3 Bälle der anderen Behandlungstypen (zusätzlich: 6/(3 Behandlungsarme-1). Entsprechend sollen es sich in der Urne jeweils 7 Bälle von dem Typ der restlichen zwei Behandlungen befinden. Die Anzahl der Bälle kann wieder direkt aus der Datenbanktabelle „responseadaptiveurn_treatmentarm“ abgelesen werden, indem die Query:

> SELECT * FROM responseadaptiveurn_treatmentarm WHERE responseadaptiveurn_id = *responseadaptiveurn_id*

ausgeführt wird.

Testfall 2: Nicht erfolgreiche Aufnahme einer neuen Antwort (Doppelte Eingabe der Antwort für einen Patienten)

Der „Prüfarzt“ wählt wieder den Menüpunkt „Antwort hinzufügen“ und gibt diesmal ID des Patienten ein, für den eine Antwort schon erfasst wurde. Eine Fehlermeldung (Abbildung 6.3.1) warnt vor der falschen Eingabe:

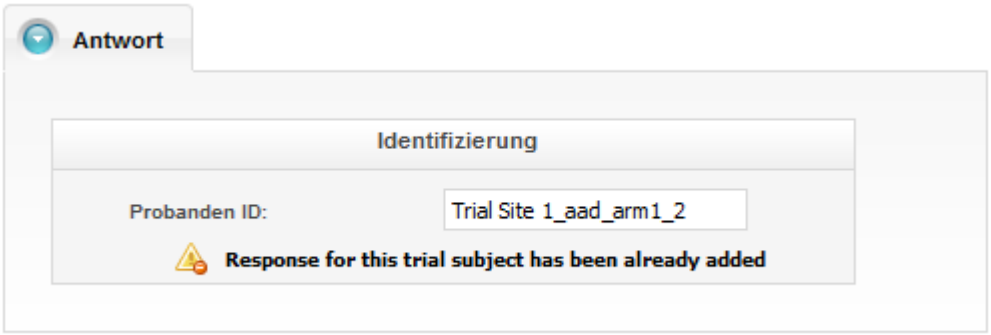

Abbildung 6.3. 1: Fehlermeldung bei der doppelten Eingabe der Probanden-ID.

7. Diskussion/Ausblick

Aus ethischen Gründen ist die Erweiterung der Anwendung RANDI2 auf response-adaptive Randomisationsverfahren sehr erwünscht. Da es viele response-adaptive Algorithmen gibt, die auf unterschiedlichen Arten funktionieren, ist diese Diplomarbeit erst der erste Schritt für die Integration von diesen Methoden in RANDI2. Aus allen zur Diskussion stehenden Verfahren wurde die „Stratifizierte und randomiesierte play-the-Winner (SRPW) Regel“ (Kapitel 2.3.4) ausgewählt und implementiert. Dieser Algorithmus bietet die Möglichkeit mehr als zwei Therapien gleichzeitig zu vergleichen und benötigt nicht sofort ein Ergebnis der Therapie des Patienten, sondern funktioniert auch mit verzögerten Antworten (was in der Realität meistens der Fall ist). Außerdem sind damit die Therapien bezüglich der wichtigen prognostischen Faktoren vergleichbar, was die Studien aussagekräftiger macht.

Das Ergebnis dieser Diplomarbeit ist die Erweiterung der Anwendung RANDI2 um das response-adaptiven Randomisationsverfahren, das auf Grundlage der SRPW Regel implementiert ist. Es bietet sich eine neue Möglichkeit, eine response-adaptive Studie zu erstellen. Anschließend ist es möglich die Patienten zu dieser Studie zu randomisieren und alle inzwischen bekannte Ergebnisse der Behandlung aufzunehmen. Die nachfolgend randomisierte Patienten haben höhere Chancen die Behandlung zu bekommen, die bis zur Randomistionsmoment bessere Ergebnisse erzielt hat.

Die SRPW Regel kann nur dichotome Antworten auswerten und entsprechend Patienten randomisieren. Da es in der Praxis oft vorkommen kann, dass mehr als zwei Antwortmöglichkeiten vorhanden sind, ist es momentan nicht möglich, jede beliebige response-adaptive Studie mit Hilfe von „RANDI2“ zu randomisieren. Deswegen ist es zu empfehlen, die Anwendung mit anderen response-adaptiven Randomisationsalgorithmen zu erweitern, die zusätzlich mit unterschiedlichen Antworttypen umgehen können.

Es ist im Rahmen dieser Arbeit nicht untersucht worden, wie das Verfahren mit der Zeitverzögerung funktioniert. Die Frage, in wie weit die Zeitverzögerung bei der Antworteingabe die Patientenzuordnung und die Studienergebnisse beeinflusst, bleibt offen. Als eine der Möglichkeiten, um diese Frage zu beantworten, bietet sich an die Simulationsmodul zu erweitern. Die Erweiterung kann sowohl die Randomisation der Patienten als auch die Bekanntgabe der Rückantwort mit der

Zeitverzögerung simulieren, um das Verhalten des Algorithmus beobachten zu können.

8.Literaturverzeichnis

[1] M. Schumacher, G. Schulgen/Kristiansen: *Methodik klinischer Studien: Methodische Grundlagen der Planung, Durchführung und Auswertung*, Zweite, überarbeitete und erweiterte Auflage, Springer-Verlag, Berlin Heidelberg; 2007. S.195-205.

[2] E. M. Connor, M.D., R. S. Sperling, et al.: *Reduction of maternal-infant transmission of human immunodeficiency virus type 1 with zidovudine treatment*, The New England Journal of Medicine, 3.November; 1994.

[3] J. Loeliger: *Version Control with Git*, O'Reilly Media, 2009

[4] Git Homepage, URL: *http://git-scm.com/* (7. März 2011)

[5] Git Wiki: URL: *https://git.wiki.kernel.org/index.php/Main_Page*. Stand: 24. Feb. 2011(abgerufen am 7. März 2011)

[6] Wikipedia: URL: *http://de.wikipedia.org/wiki/ICEfaces.* Stand: 26. Dec. 2010 (abgerufen am 13. Juli 2011)

[7] URL: *http://christopher.over-blog.de/article-23495029.html* (abgerufen am 14.07.2011)

[8] Wikipedia: URL: *http://de.wikipedia.org/wiki/Spring_(Framework)*.

Stand: 13.Mai 2011 (abgerufen am 14.07.2011)

[9] URL: *http://www.springsource.org/about* (abgerufen am 14.07.2011)

[10] D. Schrimpf*: Umsetzung und Simulation ausgewählter Randomisationverfahren bei klinischen Studien in der Webanwendung RANDI2.* Diplomarbeit im Studiengang Medizinische Informatik, Heidelberg, 2010

[11] C. Roberts, D. Torgerson: *Randomisation methods in clinical trials*, British Medical Journal 1998; 317:1301

[12] W. F. Rosenberger, J. M. Lachin: *Randomisation in clinical trials. Theory and Practice*; 2002; S.1-14

[13] F. Hu, W. F. Rosenberger: *The Theory of Response-Adaptive Randomization in Clinical Trials,* John Wiley & Sons, Inc., Hoboken, New Jersey; 2006, S.1-10.

[14] B. F. Hu, L. X. Zhang: *Asymptotic Properties of Doubly Adaptive Biased Coin Designs for Multitreatment Clinical Trials*, The Annals of Statistics; 2004, Vol. 32, No.1, S. 268-301

[15] W. F. Rosenberger, M. Lachin: *The Use of Response-Adaptive Designs in Clinical Trials,* Controlled clinical trials 14(6):471-84, Dec. 1993

[16] M. Zelen: *Play the winner rule and the controlled clinical trial,* Journal of the American Statistical Association; März 1969, S.131-146

[17] L. J. Wei, S. Duram: *The randomized play-the-winner rule in medical trials,* Journal of the American Statistical Association; December 1978

[18] Y. Liang, K. C. Carriere: *Stratified and randomized play-the-winner rule*, Statistical Methods in Medical Research; 2008; Vol. 17 No. 6; 581-593

[19] F. Hu, L. X. Zhang, S. H. Cheung, W. S. Chan: *Doubly adaptive biased coin designs with delayed responses*, The Canadian Journal of Statistics*,* Vol. 34, No. 4; 2008, p. 541-559

[20] L. X. Zhang: *Adaptive Allocation Theory in Clinical Trials*, arXiv:math/0612811v1 [math.ST]; 28 Dec. 2006

[21] R. Eisele, M. B. Woodroofe: *Central Limit Theorems for Doubly Adaptive Biased Coin Designs*, The Annals of Statistics; 1995, Vol.23, No. 1, S. 234-254